
Universidad Carlos III de Madrid
Escuela Politécnica Superior

Departamento de Informática
Doctorado en Ciencia y Tecnología Informática

Tesis Doctoral

# Arquitectura Para la Gestión y Coordinación de Sistemas Multisensor

Álvaro Luis Bustamante

Dirigida por

José Manuel Molina López

Miguel Ángel Patricio Guisado

Junio de 2015

E-mail: aluis@inf.uc3m.es

Telephone: +34 91 856 1320

Address:

Grupo de Inteligencia Artificial Aplicada
Departamento de Informática
Universidad Carlos III de Madrid
Av. de la Universidad Carlos III, 22
Colmenarejo 28270 — Spain

# Arquitectura Para la Gestión y Coordinación de Sistemas Multisensor

**Autor**: Alvaro Luis Bustamante
**Directores**: José Manuel Molina López
Miguel Angel Patricio Guisado

Firma del Tribunal Calificador:

| | Nombre y Apellidos | Firma |
|---|---|---|
| Presidente: D. | ..... | ..... |
| Vocal: D. | ..... | ..... |
| Secretario: D. | ..... | ..... |

Calificación: .....

Colmenarejo, ..... de ..... de 2015.

A mi familia.

# Índice general

# Índice de figuras

# Índice de tablas

# Resumen

HOY en día podemos encontrarnos con numerosos entornos que cuentan con un gran conjunto de sensores heterogéneos espacialmente distribuidos. Entornos que pueden ir desde sistemas de vigilancia que incluyen un despliegue de múltiples sensores como cámaras o sensores de localización, a entornos de monitorización de procesos industriales, donde se supervisa cada uno de los procesos para comprobar el correcto funcionamiento del sistema. Inclusive actualmente podemos hablar de términos más modernos como el de Smart Cities que permiten la optimización de los recursos de una ciudad a través de su constante monitorización. En todos estos entornos es común encontrarnos con multitud de sensores percibiendo información del entorno.

Estos entornos son de indudable utilidad en cada uno de sus ámbitos, y según avanza la tecnología podemos mejorarlos con la integración de más y mejores sensores. Esto resulta de utilidad porque nos permite monitorizar con más precisión y calidad las diferentes entidades o parámetros del entorno. Sin embargo esto puede suponer un problema a la hora de gestionar y coordinar diferentes tareas sobre el entorno, debido principalmente a la gran cantidad de información generada que tiene que ser analizada. En este sentido, la fusión de datos se volverá un aliado indispensable para mejorar los procesos de análisis y toma de decisiones.

En estos entornos además podemos encontrarnos con sensores u actuadores que pueden requerir de cierta gestión, como por ejemplo el manejo de una cámara PTZ para monitorizar regiones o entidades de interés del entorno. Esta gestión se puede hacer de manera manual a través de un operador humano, pero puede convertirse en una tarea compleja cuando contamos con multitud de sensores similares que puedan requerir una gestión coordinada.

En este sentido, en la tesis se abordará este problema de gestión y coordinación desde el punto de vista de un sistema multi-agente. Para ello diseñaremos una arquitectura general, que podrá ser aplicada a diferentes casos de uso. Sobre este diseño, evaluaremos su aplicación a dos entornos diferentes. El primer entorno consistirá en un sistema de vigilancia multi-cámara, donde será necesario realizar un control autónomo de las cámaras PTZ para monitorizar las diferentes entidades de interés. El segundo entorno se desarrollará sobre un sistema de vigilancia marítima, donde además de las cámaras podremos contar con otro tipo de sensores como radares o estaciones AIS.

La experimentación llevada a cabo en esta tesis a través de la aplicación multi-cámara, ha permitido evaluar la arquitectura desde el punto de vista de la integración en un entorno

real. La aplicación sobre el entorno marítimo, en este caso desarrollado sobre un entorno simulado, nos ha permitido además evaluar la adecuación de de la arquitectura a diferentes sensores e información de las entidades. En ambos casos hemos podido observar la utilidad del sistema multi-agente desarrollado, así como su adecuación a este tipo de entornos, que dada su naturaleza son inherentemente distribuidos.

# Abstract

NOWADAYS it is common to find several heterogeneous sensors spatially distributed in environments. Environments that can range from surveillance systems that include a set of multiple sensors such as cameras or location sensors. To environments monitoring industrial processes, where it is important to monitor several processes to ensure a proper system operation. Moreover we can talk about more modern terms as Smart Cities, which allow optimizing the city resources through its constant monitoring. Thus, in all those deployments, it is common to find multiple sensors perceiving information from the environment.

These environments are undoubtedly useful in each of their fields, and as technology advances, we can improve the integration of more and better sensors. This is useful because it allows a better and more accurately monitoring of the different entities or environmental parameters. However this can be a problem when it comes to managing and coordinating different tasks over the environment, mainly due to the large amount of information generated required to be analyzed. In this sense, data fusion will become an indispensable ally to improve the analysis and decision making processes.

Moreover, in these environments we can find multiple sensors or actuators that may require some kind of management, such as managing a PTZ camera to monitor regions or entities of interest. This management can be done manually by a human operator, but can become a complex task when we have multiple sensors requiring a coordinated management.

In this sense, this thesis tackle the problem of multi-sensor management and coordination from the point of view of a multi-agent system. It is proposed a general multi-agent design that can be applied to different use cases. This design will be evaluated in two different environments. The first one will consist on a multi-camera surveillance system, where the multi-agent system must achieve an autonomous control of PTZ cameras to monitor different entities of interest. The second environment will rely on a maritime surveillance system, where in addition to the cameras, we will introduce other sensor types such as radar or AIS stations.

Some experiments were carried out to evaluate the multi-agent system designed. The experiments done in the multi-camera application, allowed us to evaluate the architecture from the point of view of its integration in a real environment. Meanwhile the application in the maritime environment, developed over a simulated environment, let us evaluate the suitability of the architecture to different sensors and information. In both cases we have been able

to observe the usefulness of the developed multi-agent system and its adaptation to these environments inherently distributed.

# Agradecimientos

CUANDO uno se embarca en el desarrollo de una tesis, nunca se puede hacer a la idea de la cantidad de personas que directa o indirectamente se verán involucradas. No es un trabajo en solitario, si no un largo camino donde aprendes de unos y de otros. Este camino empieza con mis directores de tesis José Manuel y Miguel Ángel. Sin duda gracias por haberme apoyado desde el principio con el máster, motivarme de vez en cuando para acabar la tesis de una vez, y haber confiando en mí para desarrollar un proyecto detrás de otro.

También agradecerle a Jesús García la inestimable experiencia, conocimiento, y tiempo compartido a lo largo de los proyectos que hemos desarrollado, con los que he podido aprender un montón de cosas. A Berlanga por sus charlas interesantes, su análisis crítico, y las horas de Pádel de los últimos años en los que hemos jugado como equipo. ¡Seguro que en los próximos partidos conseguimos mejores resultados!.

Especial recuerdo guardaré de los últimos años con Ramón y mi compañero de fatigas Kike, con nuestra hora del robo obligada, en las que siempre acabábamos hablando de comida, o rememorando noches épicas. También gracias a la panda de delincuentes del GIAA: Gonzalo, Mirren, El Marqués, Albertito, Rodri, Luis, Nayat, Juanito, y Vero, con los que he compartido grandes momentos, como esos congresos tan surrealistas en Salamanca. También agradecer a Carbó, Eli, Juanita, e Irene las interesantes charlas compartidas en las comidas, y por qué no a Carmela, Doris, y Anabel, que tantos y tantos años nos han atendido en la cafetería.

A mi familia en general, los tíos, los primos, y los abuelos, por estar ahí siempre que se los necesita. En especial a mi padre por haberme apoyado durante tantos años, y por haberme dicho más de una vez "¿por qué no sigues estudiando un poco más?". A mi madre por apoyar con los ojos cerrados todas las decisiones que he tomado. A mi hermana por haber sido la mejor amiga durante tantos años, aunque siempre hemos tenido nuestro más y nuestros menos. A mis tíos Juan y Susana, que me aguantaron durante casi tres años en su casa como si fuera su hijo. A mi abuelo Ángel, allí donde estés, que siempre que hablaba de mí decía con una sonrisa en la cara que era "Doctor Ingeniero", cuando apenas había comenzado. A mi abuela María por ser la consejera de la familia, y por llenar los congeladores de tuppers de comida!

Por supuesto agradecer también a mis amigos, tanto los viejos como los nuevos que he podido hacer en los últimos años y con los que he podido compartir esas barbacoas, bodas, y juegos frikis. Gracias Alberto, Moi, Sarita, Moska, Cris, Oscar, Silvi, Cesáreo, Celia, Edu, María, Dani, Ele, y los que me deje por el camino!. Y como no, agradecer a Noelia su compañía

durante estos años y por no quejarse demasiado en los últimos meses cuando sólo había tesis. ¡Te quiero Paquitori!.

El camino no acaba aquí, de hecho sólo acaba de comenzar, así que gracias a todos, porque de una forma u otra habéis contribuido a desarrollar esta tesis y a hacer el comienzo del camino más llevadero.

*Álvaro Luis Bustamante*
*Colmenarejo, Mayo de 2015.*

# 1

# Introducción

## 1.1 Contexto del problema y motivación

La Inteligencia Artificial ha sido desde mediados del siglo XX una de las disciplinas más extendidas y estudiadas dentro del ámbito de la informática. En 1950, Alan Turing se preguntó si las máquinas serían capaces de pensar, y desde entonces, no han parado de nacer nuevas disciplinas de investigación dentro de la ciencias de la computación. En gran parte, el objetivo de estas disciplinas es intenta dotar a las máquinas de un comportamiento racional que permita un comportamiento similar al de los humanos, que nos imiten, que se comporten de manera inteligente. Esta inquietud sobre las máquinas, llevo a Alan Turing a diseñar un experimento que permitiría determinar si una máquina se podría considerar inteligente o no. Para superar la prueba de Turing, la máquina debía ser lo suficientemente inteligente como para responder una serie de preguntas hechas por un humano, y que éste no fuera capaz de diferenciar si las respuestas venían de otro humano o una máquina. Por aquél entonces se pensaba que sería posible conseguir superar ésta y otras pruebas a finales del siglo XX. Sin embargo, aún hoy en día es difícil encontrar sistemas que sean medianamente capaces de imitar nuestro comportamiento e interaccionar correctamente con nuestro entorno.

Es precisamente la interacción con nuestro entorno uno de los campos que más problemas ha planteado dentro de la inteligencia artificial. Si bien es cierto que los ordenadores tienen una gran capacidad de cómputo que les permite procesar automáticamente grandes cantidades de información, es en el entendimiento y procesado de la información que manejamos en nuestro entorno donde fallan. Sin esta capacidad básica que cualquier humano adquiere a través de los sentidos durante los primeros años de vida, es complicado que se puedan llegar a lograr sistemas, que a nuestros ojos, realicen actividades inteligentes en nuestro entorno. Como bien dijo Kant, «todo nuestro conocimiento comienza por los sentidos, pasa de estos al entendimiento y termina en la razón».

Uno de los campos que posiblemente más repercusión y utilidad práctica ha tenido en la

Inteligencia Artificial desde sus orígenes ha sido precisamente el encargado de procesar uno de los sentidos más importantes: la vista. La visión por computador se encarga de proporcionar a los ordenadores la capacidad de entender una escena a través de las características que puede obtener de una imagen. Tras más de 40 años de investigación en este ámbito han surgido interesantes métodos que permiten el reconocimiento de objetos en función de su apariencia, el reconocimiento facial, el seguimiento de objetos, la navegación autónoma utilizando información visual, e inclusive el reconocimiento de actividades humanas, entre otros.

Además, en los últimos años hemos podido observar un avance significativo en la capacidad para desarrollar dispositivos conectados, como sensores, y actuadores. Estos dispositivos los podemos encontrar desplegados espacialmente por el entorno para realizar diferentes tareas de monitorización y control. Realizar la integración, gestión, y coordinación de este tipo de entornos supone un desafío significativo. En parte debido a la gran cantidad de elementos que podemos encontrar distribuidos espacialmente, la información que generan, y las características específicas que cada uno puede presentar.

Este gran abanico de monitorización de los entornos se aplica con éxito hoy día en entornos como sistemas de monitorización de procesos industriales, sistemas de vigilancia, como los entornos marítimos, costeros y de fronteras, e inclusive sobre entornos que están tomando relevancia en los últimos años como las Smart Cities. Muchos de estos elementos tienen que ser integrados y gestionados para que resulten de utilidad. Normalmente encontraremos a un operador humano que supervisa la información y toma las decisiones oportunas para cumplir con su cometido. Sin embargo, esto es cada vez más complicado, ya que la cantidad de información que somos capaces de generar hoy día no es comparable siquiera a la de hace unos cuantos años. Esto hace que las tareas de gestión realizadas por humanos sean cada vez menos eficientes o inclusive impracticables.

Gracias a la mezcla de diferentes disciplinas, entre las que se encuentra la inteligencia artificial, podríamos ser capaces de diseñar un sistema que alivie el trabajo del operador y permita la gestión de entornos de cierta envergadura. Por ejemplo, podríamos considerar un sistema de vigilancia costera como un puerto, donde suele haber un centro de control y diversas cámaras utilizadas por los operadores para controlar los barcos que entran y salen del puerto. También podríamos encontrar otros sensores adicionales como radares costeros, o sistemas de identificación automático (AIS), que ayudan a los operadores a visualizar la posición en tiempo real de todos los barcos. En este entorno podríamos considerar un sistema más avanzado que permita al operador automatizar ciertas tareas. Por ejemplo, un sistema que gestione automáticamente las cámaras para vigilar a los barcos que se aproximen al puerto, o barcos que vayan demasiado rápido, o simplemente que maximice el número de barcos vistos en cada momento. Se podrían diseñar muchos casos de uso que podrían ser útiles para un operador, y que pueden ser automatizados por un sistema autónomo que aproveche la información reportada por los múltiples sensores del entorno.

La propuesta de esta tesis se enfoca por tanto en el diseño de una arquitectura para la gestión y coordinación de un conjunto de sensores heterogéneos espacialmente distribuidos por el entorno. Esta arquitectura debería ser capaz de gestionar y coordinar los recursos disponibles, así como atender a las preferencias o tareas asignadas por un eventual operador. Para este diseño será necesario recurrir a múltiples técnicas que van desde los aspectos de más bajo nivel como puede ser la adquisición y tratamiento de la información, a los de más alto nivel que razonan y toman decisiones en base a estas percepciones. Este tipo de arquitectura podría tener múltiples aplicaciones en entornos reales, sobre todo en entornos de vigilancia como puertos, aeropuertos, bases militares, grandes superficies, o cualquier otro entorno que necesite ser monitorizado en tiempo real para prevenir situaciones indeseadas.

## 1.2 Objetivos

En esta tesis se pretende diseñar y evaluar una arquitectura que permita la gestión y coordinación de un conjunto de sensores heterogéneos espacialmente distribuidos por el entorno. Debido a la gran cantidad de información que este tipo de entornos puede llegar a generar, resulta indispensable aplicar técnicas de fusión de datos que nos permitan integrar la información multi-sensor y nos faciliten de algún modo los procesos de evaluación del entorno y la toma de decisiones. Por lo que la arquitectura de gestión multi-sensor se basará en modelos de fusión existentes que definan los diferentes niveles de integración de información y gestión de sensores.

Sin embargo, los modelos de fusión son diseños teóricos que contemplan las diferentes etapas o procesos involucrados en la fusión de información. Pero no especifican las arquitecturas sobre las que desarrollarlas, o cómo llevar a cabo los procesos de gestión. De este modo, en esta tesis, se plantea el diseño de la arquitectura desde el punto de vista de los sistemas multi-agente. Un sistema multi-agente trataría básicamente de un conjunto de entidades que pueden encontrarse distribuidas por el entorno, y que tienen ciertas habilidades para percibir información, razonar, comunicarse, y actuar. Esto nos permitirá diseñar una arquitectura distribuida, que encaja de forma natural con el tipo de entorno que vamos a gestionar. Para el desarrollo de esta arquitectura nos basaremos en el modelo de fusión JDL (Liggins II et al., 2008).

Se realizará un diseño general del sistema multi-agente que nos permita definir los diferentes agentes a considerar en su integración en entornos multi-sensor. Para ello se describirá el rol que desempeña cada uno de ellos, qué información gestionan, cómo se coordinarán, etc. Este diseño, aún de alto nivel, servirá como base para su aplicación a entornos más especializados. Ya que en este nivel aún no consideramos el uso de ningún sensor en particular, o el cumplimiento de ningún objetivo. Esto sería dependiente de cada entorno, y sería necesario especializar el diseño proporcionado para los diferentes entornos donde lo integremos.

Por este motivo, además de la propuesta general del sistema multi-agente, vamos a considerar su aplicación a dos entornos de vigilancia diferentes. Esto nos permitirá evaluar la adecuación de la arquitectura multi-agente sobre los sistemas multi-sensor, así como proporcionar un diseño de referencia que sirva de guía para la integración de la arquitectura sobre diferentes entornos.

En la sección 1.2.1 se describen los objetivos específicos a realizar dentro de la definición de la arquitectura multi-agente. Por otro lado, los objetivos de las dos aplicaciones realizadas sobre la arquitectura se describen en más detalle en las secciones 1.2.2.1 y 1.2.2.2.

### 1.2.1 Arquitectura general para la gestión y coordinación de entornos multi-sensor

El objetivo principal consistirá en diseñar un sistema multi-agente que permita la gestión de sensores u actuadores heterogéneos en función de las percepciones de su entorno. El sistema

diseñado debe ser capaz de explotar esta información para realizar una gestión autónoma en función de las preferencias establecidas por el operador. Se utilizarán los agentes por su capacidad para percibir información sobre el entorno, comunicarse para la coordinación, y razonar sobre el estado del entorno.

El diseño de este tipo de sistemas cuenta con numerosas tareas a realizar. Entre ellas podremos enumerar las siguientes:

**Definición de agentes:**

Las entidades básicas en un sistema multi-agente son los llamados agentes. Son entidades independientes que pueden tener habilidades para interaccionar con su entorno y con otros agentes. En los entornos con los que vamos a trabajar podríamos encontrar numerosos agentes que representan diferentes entidades, como los sensores, actuadores, el operador humano, un sistema de fusión, etc. Por lo tanto será indispensable poder identificar los agentes que necesitaremos. Cada uno de los agentes identificados estará vinculado con una tarea específica. Debido a la naturaleza distribuida del entorno, algunos agentes podrían estar más relacionados con la interacción de los sensores a bajo nivel, mientras que otros agentes podrían estar relacionados con las lógicas de gestión de más alto nivel. De este modo, se describirá el propósito de cada agente dentro del diseño.

**Definición de las relaciones:**

Los agentes como entidades independientes no tendrían sentido dentro de un entorno distribuido donde es necesaria la coordinación para realizar una gestión eficiente. Es necesario que estos agentes se apoyen en otros agentes para llevar a cabo los propósitos para los que han sido diseñados. Por lo que habrá que describir cómo y cuándo interaccionan los agentes.

**Definición de la comunicación:**

Todos los agentes que cooperan entre sí necesitan comunicarse de algún modo, y por tanto pueden surgir diferentes flujos de información. Podríamos encontrarnos con la información proporcionada por los sensores, los flujos de control de los sensores, los mensajes intercambiados para colaborar y razonar, etc. Todos estos flujos de información deben ser definidos con el fin de que los diferentes agentes sean capaces de interaccionar e interactuar con el entorno. Para ello se describirá la información básica de entrada y de salida que se espera para cada agente.

**Mecanismos de coordinación:**

Una vez definidos los agentes, sus relaciones, la comunicación entre ellos, etc., se tendría que definir cómo se va a procesar la información en cada uno de los agentes, cómo van a razonar, y cómo y cuándo comenzarán los mecanismos de colaboración para resolver problemas en grupo. Esta definición sería dependiente de cada entorno y de los objetivos

particulares a solucionar. Sin embargo es necesario contemplar el despliegue de estos mecanismos a la hora de diseñar el sistema multi-agente.

### 1.2.2 Desarrollo de aplicación MAS sobre entornos de vigilancia

Una vez diseñado el planteamiento general de la arquitectura multi-agente, será de utilidad poder aplicar este modelo teórico a diferentes entornos específicos de gestión multi-sensor. Esto nos permitirá validar el diseño realizado, así como proporcionar un diseño de referencia que permita su aplicación a diferentes entornos.

Para evaluar la propuesta del diseño multi-agente, se buscará su aplicación a dos entornos de vigilancia diferentes. El primero consistirá en la adaptación y evaluación del diseño multi-agente a un entorno de vigilancia multi-cámara. En este entorno contaremos únicamente con sensores de visión que necesitan gestión y coordinación para realizar tareas de monitorización autónomas. Estas cámaras tienen la particularidad de que pueden ser utilizadas para realizar determinadas tareas de seguimiento, ya que permiten modificar su campo de visión. Estas se conocen como cámaras PTZ, y permiten ajustar la rotación de la cámara para observar determinados elementos de interés del entorno.

Este tipo de sensores los podemos encontrar hoy día en múltiples sistemas de seguridad basados en cámaras de vigilancia, como el que podemos encontrar en el laboratorio de investigación del GIAA (Grupo de Inteligencia Artificial Aplicada). Trabajar con un entorno y una implementación real, nos permitirá enfrentarnos a múltiples desafíos prácticos que posiblemente no se habían contemplado en el diseño original. Principalmente porque cuando se realiza un diseño de alto nivel se tiende a relajar la especificación con el fin de generalizar lo máximo posible su aplicación a diferentes casos de uso. Es en la particularización de la aplicación cuando surgen problemas y retos a resolver que no estaban previstos. Por este motivo resultará útil la integración del diseño general en diferentes entornos reales.

Sin embargo, un entorno moderno de vigilancia no trata sólo con cámaras para monitorizar el entorno. Los más modernos pueden incorporar múltiples sistemas de localización, tanto en interiores, como en exteriores. Toda esta información puede ser también útil en el manejo y control de las cámaras, ya que se trata de información adicional del entorno que puede mejorar el rendimiento del sistema si lo comparamos con un sistema tradicional basado únicamente en sistemas de visión. La gran ventaja que presentaría la utilización de este tipo de sensores, dependiendo de la tecnología, es que podríamos contar con la identificación de objetivos, tener monitorizados elementos que las cámaras no están viendo, o inclusive no pueden ver por las oclusiones, tener más precisión en la información del estado, etc.

Por lo que además de la aplicación sobre el entorno multi-cámara, propondremos la aplicación de la arquitectura sobre un entorno de vigilancia marítimo, que cuenta con más variedad de

sensores que proporcionan información complementaria sobre el entorno. Estos sensores serán radares y estaciones AIS que estarán detectando las embarcaciones presentes en un entorno marítimo. Este entorno también contará con cámaras PTZ que serán el elemento principal a coordinar, de manera similar la aplicación multi-cámara.

Estos dos entornos particulares se describen en las siguientes secciones.

#### 1.2.2.1 Aplicación en entorno de vigilancia multi-cámara

Este entorno tratará únicamente con la gestión de sensores de visión (cámaras PTZ) para realizar determinadas tareas de monitorización autónoma. Este tipo de entornos es muy común en los sistemas de videovigilancia actuales, ya que permiten un despliegue de sensores relativamente barato y pueden proporcionar información visual del entorno muy valiosa. Sin embargo, el uso de estos sensores de visión añade determinadas dificultades cuando se integran en sistemas automáticos que procesan la información y realizan su gestión.

La primera dificultad que presentan es la cantidad de información que son capaces de generar. Dada su naturaleza, se trata de sensores que capturan instantáneas del entorno a razón de unos 25 fotogramas por segundo (dependiendo de la cámara). Esto complica las tareas de análisis de la información, además de su transmisión a las diferentes partes de la arquitectura. Por ejemplo, sería útil poder transmitir la secuencia de vídeo al centro de mando donde se encuentra el operador. Pero también podría ser útil utilizar esta información para almacenarla para su posterior análisis. Inclusive podría ser necesario que esta información pudiera ser monitorizada de forma remota a la instalación. Y ya no solo eso, podríamos necesitar realizar un procesamiento sobre las imágenes capturadas para detectar posibles elementos de interés, como podría ser aplicar técnicas de visión artificial para detectar personas, sus actividades, su ubicación, etc. Y a su vez esta información podría ser útil para el operador o para registrar una sucesión de eventos en el entorno. Gestionar todos estos flujos de información, junto con la aplicación de algoritmos de visión e integración con el sistema de vigilancia, ya supone realizar un gran esfuerzo.

La segunda dificultad consistiría en que no son sensores pasivos que se limitan a capturar imágenes del entorno. Las cámaras PTZ pueden ser controladas para cambiar su campo de visión y así poder monitorizar los diferentes objetivos o áreas de interés. Esto añade una necesidad dentro de la arquitectura, y es que será necesario gestionarlas para obtener un mejor rendimiento en la tarea de vigilancia. Pero ya no sólo será necesario gestionarlas por un operador humano, debemos poder contemplar un control autónomo como el que proponemos con nuestra arquitectura multi-agente. Esto tiene su dificultad añadida porque una cámara PTZ normalmente cuenta con una única interfaz de control, que debería ser compartida entre la gestión autónoma y el operador.

Por lo tanto, podemos ver que la aplicación del diseño multi-agente sobre el entorno multi-cámara no es una tarea trivial, al margen de tener que especificar y adecuar el diseño para este caso particular. De esta forma, la integración de este tipo de sensores dentro de la arquitectura multi-agente será una prioridad y uno de los objetivos a resolver si queremos realizar una integración satisfactoria. La integración de este sensor en la arquitectura multi-agente incluirá algunos objetivos específicos para estos sensores, que podemos resumir a continuación:

**Captura y transmisión de la imagen:**

El acceso físico a una cámara suele estar limitado, y muchas veces sólo es posible acceder desde una única interfaz física. Esto complica que una imagen capturada pueda ser empleada por varios procesos simultáneamente. Habrá que considerar un sistema que permita capturar y transmitir la imagen a diferentes procesos locales o remotos. Por ejemplo, la imagen suele ser útil para la visualización del entorno, la grabación para su posterior análisis, y en un sistema automático como el que se propone, para ser analizada en tiempo real y percibir la información del entorno. Además, la codificación y transmisión de la imagen deberá seguir un estándar para que todos los elementos implicados en el sistema de vigilancia puedan interoperar correctamente.

**Control de la cámara:**

Al igual que con la captura de la imagen, la interfaz de la cámara suele proporcionar un acceso exclusivo a su interfaz de control. En un sistema automático de vigilancia habrá procesos controlando la cámara automáticamente dependiendo de los objetivos establecidos. Eventualmente también se podría requerir la intervención de un operador que desee obtener más detalle sobre un objetivo en concreto. Esto implica que más de dos procesos podrían requerir el control de la cámara, lo que hace necesario implementar una interfaz común que lo permita.

**Procesado de la imagen:**

Adicionalmente podríamos considerar el procesado de la imagen como un elemento más a tratar en la integración del sensor. Si pensamos en que habrá un sistema autónomo que debería estar analizando las imágenes para realizar el control automático sobre las cámaras, es obvio que este proceso tiene que ser lo más rápido posible. Si este análisis lo realizamos en el sistema autónomo perderemos tiempo codificando, transmitiendo y decodificando la imagen. Si además varios sistemas o componentes en paralelo pueden requerir el mismo análisis de imagen, estaríamos procesando y transmitiendo información por duplicado, perdiendo así ancho de banda, capacidad de procesado, y obtendríamos una respuesta más lenta. Por estos motivos podría ser conveniente realizar este análisis de manera local a la cámara, y transmitir únicamente el resultado del análisis al resto de procesos o sistemas. Por ejemplo, sería posible aplicar algoritmos de seguimiento en cada cámara, y enviar al sistema autónomo únicamente esta información para que realicen el

control, ahorrándose la transmisión del vídeo. Inclusive, si el operador recibe un flujo de vídeo, y una serie de metadatos asociados procedentes del análisis de las imágenes, se podría mostrar información superpuesta como los objetivos detectados, su identificador, el tiempo de vida, la trayectoria descrita, etc.

Podemos observar una representación de las diferentes necesidades de integración de la cámara PTZ en la figura 1.1. En esta podemos observar diferentes elementos recibiendo la transmisión de información, como el sistema de visualización, un grabador de vídeo, los algoritmos de visión por computador, etc. También podemos observar un flujo de control desde el sistema autónomo, que en nuestro caso será el sistema multi-agente, así como del operador que supervisa el entorno.

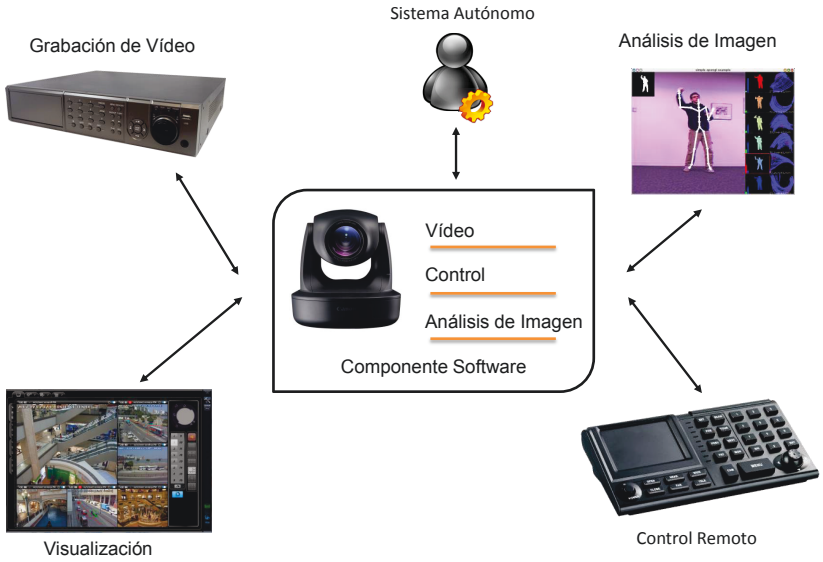


Figura 1.1: Complejidad de integración de una cámara de vigilancia PTZ. La imagen proporcionada podría ser explotada por múltiples sistemas, como algoritmos de visión, monitorización remota, o sistemas de almacenamiento. Estas cámaras además deberán permitir su gestión, ya sea a través del operador o un sistema autónomo.

Por lo tanto, sobre este entorno y sus particularidades de integración, aplicaremos el diseño del sistema multi-agente que permita la gestión autónoma de las cámaras PTZ atendiendo a los objetivos establecidos por el operador. En nuestro caso consistirán en la monitorización autónoma de las diferentes entidades detectadas en el entorno. Para ello tendremos que identificar los agentes necesarios para llevar a cabo esta tarea, así como concretar la información de entrada/salida de cada uno de ellos, la relación de los diferentes agentes instanciados, y los mecanismos de coordinación a utilizar para seleccionar los objetivos a monitorizar. Para esta aplicación en concreto trabajaremos sobre el laboratorio del GIAA, que cuenta con un conjunto de cámaras y servidores que nos permitirán desarrollar la aplicación sobre un escenario real.

Esta especificación será muy similar a la que hemos definido en la sección 1.2.1, pero estará

particularizada al problema concreto de la gestión multi-cámara. Esta especificación debería servir para evaluar la adecuación del diseño general a un entorno específico, pero no pretenderá especificar un diseño definitivo para resolver todas las tareas de gestión multi-cámara.

#### 1.2.2.2 Aplicación en entorno de vigilancia marítimo

El objetivo de esta aplicación consistirá en la aplicación del diseño del sistema multi-agente al entorno particular de la vigilancia marítima. En este entorno deberá realizar el control autónomo de las cámaras PTZ para el seguimiento de determinados objetivos en función de las detecciones generadas por los sensores. Igual que en la aplicación multi-cámara, esta aplicación permitirá evaluar la adecuación del sistema multi-agente general a un caso de uso concreto, así como discutir la integración de nuevos sensores, y la reutilización de componentes que ya hemos diseñado. Podemos ver una representación de este tipo de entornos en la figura 1.2, donde podemos observar los nuevos sensores considerados como los radares o las estaciones AIS, junto con los sensores de vídeo a gestionar.

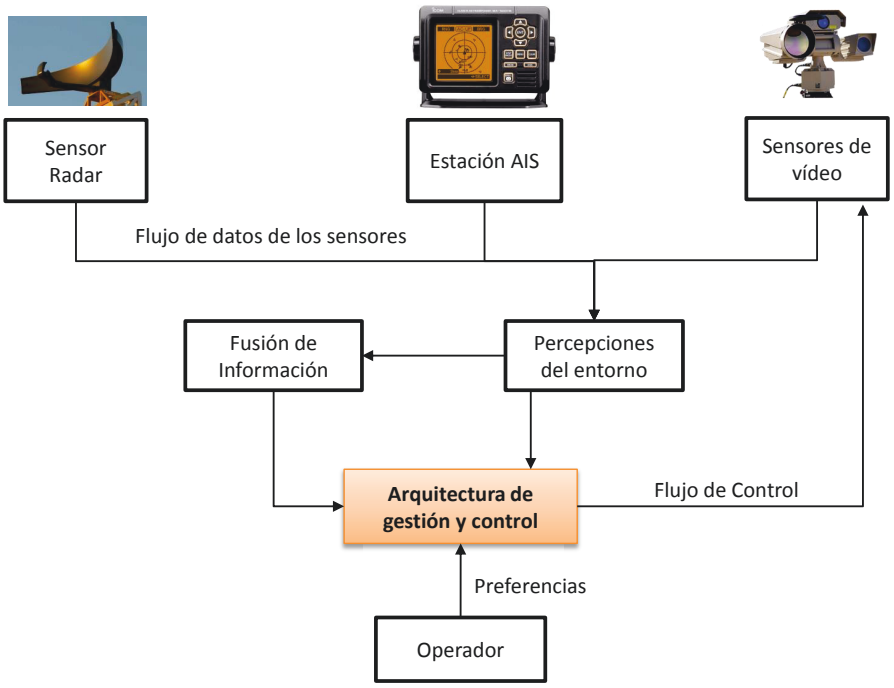


Figura 1.2: Descripción del entorno de vigilancia marítima donde se integrará el sistema multi-agente. En este contaremos con información de radares, estaciones AIS, y cámaras PTZ.

Como hemos visto en la sección 1.2.2.1, tratar de trabajar con un entorno real supone un desafío a diferentes niveles, inclusive con un entorno donde sólo necesitamos gestionar cámaras PTZ. Este entorno supone un desafío añadido, ya no sólo por la utilización de nuevos sensores como los radares o las estaciones AIS, si no por el hecho de no tener al alcance un sistema de estas características. Las empresas son muy cautelosas a la hora de permitir el acceso a

este tipo de sistemas, e inclusive a la información que puede llegar a ser capturada del entorno. Normalmente estarán atadas a los permisos o concesiones que les permita realizar el cliente. Además, estos sistemas una vez instalados se encuentran funcionando las 24 horas al día con operadores realizando las tareas de monitorización. Por lo que resultaría muy complicado desarrollar y evaluar el sistema propuesto.

En este caso, para solucionar este problema desarrollaremos además un simulador de entornos marítimos que nos permitirán integrar el diseño de nuestro sistema multi-agente y evaluar su funcionamiento. Este diseño, una vez probado y evaluado en el simulador, debería ser fácil de extrapolar al mundo real. Este simulador permitirá configurar diferentes elementos como sensores radar, estaciones AIS, y cámaras PTZ, simular diferentes embarcaciones y su dinámica de movimiento, así como simular las detecciones realizadas por cada uno de los sensores. Además permitirá el despliegue de los diferentes agentes que diseñemos para realizar las tareas de control en función de las percepciones del entorno.

## 1.3 Estructura del documento

En la sección 2 se describirá parte del estado del arte analizado para la realización de esta tesis. Se tratará con la fusión de datos y los diferentes modelos que podemos encontrar, la problemática y los enfoques de la gestión multi-sensor, y por último, las particularidades de la gestión de sensores de visión como un elemento más del entorno.

En el capítulo 3 se propone el diseño del sistema multi-agente para la gestión y coordinación de entornos multi-sensor. En este describiremos el diseño del sistema multi-agente, identificando los diferentes agentes, la información que manejan, y sus relaciones. Este diseño genérico será la base para poder desarrollar aplicaciones de gestión multi-sensor sobre entornos más concretos.

Para evaluar la adecuación del modelo teórico general sobre entornos más específicos, se describirá su aplicación a dos entornos diferente. El primero será un entorno de vigilancia multi-cámara que permite la monitorización automática de los diferentes objetivos presentes en el entorno. Esta aplicación se describe en la sección 4.

La segunda aplicación consistirá en el diseño de la arquitectura aplicado a otro entorno diferente como son los entornos de vigilancia marítima. En este entorno contaremos con diferentes sensores de monitorización como los sensores radar y las estaciones AIS, además de la cámaras que deberán ser controladas automáticamente. Este entorno quedaría descrito en la sección 5.

Para finalizar, en la sección 6 terminaremos con las conclusiones obtenidas tras la realización de la tesis, además de presentar posibles futuras líneas de investigación relacionadas.

En el apéndice A también proporcionamos algunos proyectos relacionados con las propuestas de esta tesis, así como las publicaciones que han soportado su desarrollo.

# 2

# Estado del arte

## 2.1 Introducción

Los entornos multi-sensor se están convirtiendo en sistemas sumamente importantes en una gran variedad de aplicaciones tanto militares como civiles. Debido a que un único sensor generalmente sólo puede percibir una información parcial o limitada del entorno, es deseable poder contar con múltiples sensores desplegados que permitan generar una imagen global más precisa. La información de múltiples sensores heterogéneos puede ser combinada utilizando algoritmos de fusión de datos para obtener una representación global que facilite el análisis de los datos y la toma de decisiones. Por lo tanto, los sistemas multi-sensor nos permitirán obtener información más amplia del entorno comparada con sistemas que dependen de un único sensor.

Además, podemos ver cada vez más avances en las tecnologías de monitorización, contando con sensores cada vez más configurables, y con más capacidad de generar información. Por lo tanto, la creciente sofisticación de los sensores, junto con la gran cantidad de información a procesar podría llegar a desbordar a los operadores humanos a cargo de su monitorización. Esto motiva cada vez más el interés en la gestión automática y semi-automática de los recursos de monitorización para mejorar el rendimiento general en los procesos de percepción del entorno y la fusión de datos.

La integración de múltiples sensores monitorizando el entorno en un sistema de vigilancia automático supone tener que afrontar diferentes problemas a diferentes niveles. Desde los problemas de más bajo nivel, como puede ser la adquisición, procesado, y comunicación de la información, a la fusión de datos para integrar toda la información del entorno, hasta las capas de más alto nivel que pueden implicar el control, y la coordinación de sensores (Abidi et al., 2008). En la actualidad podemos encontrar diferentes tipos de sensores de visión y localización que se despliegan normalmente en los sistemas de vigilancia. Estos sensores necesitan ser gestionados y coordinados de una forma óptima con el fin de llevar a cabo la tarea de vigilancia deseada. Los sensores de visión además necesitan ser tratados de una forma un tanto particular

debido al tipo de información que proporcionan. No son sensores normales que se conectan y comienzan a generar información directamente tratable. En estos caso es además necesario procesar la información atendiendo a diferentes particularidades que queramos extraer del entorno, por ejemplo aplicando algoritmos de visión artificial, calibrando las cámaras, orientando su campo de visión, etc.

Por lo tanto, dentro del estado del arte nos centraremos en diferentes aspectos relacionados con la gestión multi-sensor. El primero de ellos consistirá en la fusión de datos, descrita en la sección 2.2, que nos permitirá hacernos una idea de los diferentes problemas que se pueden resolver, y los modelos de fusión que podemos encontrar en la actualidad para su aplicación dentro de la gestión multi-sensor.

En la sección 2.3 se verán algunos de los problemas que plantea la gestión multi-sensor, así como los enfoques que se han llevado a cabo para su solución. En este caso trataremos con sensores genéricos que normalmente están relacionados con entornos de vigilancia o militares, que posiblemente sean los campos más aplicados de la gestión multi-sensor.

Para finalizar abordaremos el problema de gestión desde el punto de vista de los sensores de visión 2.4, ya que estos pueden llegar a presentar particularidades adicionales que merece la pena estudiar por separado, además de que son sensores con los que trabajaremos en las dos aplicaciones propuestas.

## 2.2 Fusión de datos

### 2.2.1 Introducción

El término de fusión como tal puede referirse a la combinación de dos o más elementos diferentes. Este término es ampliamente utilizado en diferentes disciplinas como la energía nuclear y la gastronomía. En este caso nos centramos en el término de la fusión desde el punto de vista de fusión de sensores. La fusión de sensores podría definirse como la combinación de la información proporcionada o derivada de múltiples sensores de tal forma que la información resultante es de algún modo mejor que si utilizásemos las fuentes de datos individualmente. El término "mejor" en este caso puede significar más preciso, más completo, más concreto (Hall and Llinas, 2001, 1997; Liggins II et al., 2008).

Los entornos que nos podemos encontrar hoy día son capaces de generar gran cantidad de información a través de los múltiples sensores distribuidos espacialmente. Cada día además contamos con mayor y mejor tecnología para monitorizar diferentes aspectos. Por ejemplo, en los entornos de vigilancia actuales podemos encontrar múltiples técnicas de localización de personas, que van desde sistemas de localización basados en GSM (Otsason et al., 2005), basados en WiFi (Evennou and Marx, 2006), RFID (Jin et al., 2006), Bluetooth (Pei et al., 2010), o un conjunto de varias a la vez (Hossain et al., 2007). Incluso en determinados entornos de vigilancia costera, aeropuertos, fronteras, etc., nos podríamos encontrar con diferentes tecnologías como radares, sistemas de identificación automática (AIS), e inclusive imágenes satelitales (Carthel et al., 2007; Schwehr and McGillivary, 2007).

Por lo tanto estamos considerando entornos que pueden tener la capacidad de generar una gran cantidad de información que debería ser integrada mediante la fusión de datos para mejorar la tarea de monitorización o vigilancia. Esto nos permitirá tener información más precisa, completa, y no redundante de los diferentes elementos detectados por los diferentes sensores. Este tipo de información resulta de gran utilidad en los procesos de toma de decisiones por operadores humanos (Nilsson, 2008). Por lo que será de vital importancia también en los procesos de gestión y coordinación autónomos.

La fusión de datos es un campo de estudio que lleva a sus espaldas décadas de investigación. En concreto, la fusión de datos en entornos multi-sensor se puso en auge sobre el año 1990 como una tecnología emergente (Liggins II et al., 2008), generando múltiples problemas relacionados con el diseño y selección de algoritmos, las arquitecturas de fusión, definición de pruebas y sistemas de evaluación, comprobación de utilidad, etc.

### 2.2.2 Aplicaciones de la fusión de datos

Uno de los campos donde más se ha empleado la fusión de datos es en los entornos de vigilancia. Estos entornos suelen contener un gran número de sensores distribuidos espacialmente por el entorno, y en muchos casos estos proporcionan información redundante sobre el entorno. Aquí surge la necesidad de la fusión de datos para poder obtener una imagen global del estado del entorno, así como para mejorar la precisión de los diferentes objetivos monitorizados. Estos entornos pueden ser aeropuertos, costas, puertos marítimos, fronteras, etc.

Podemos encontrar diferentes trabajos enfocados a realizar fusión de sensores en entornos marítimos, como por ejemplo el desarrollado por (Carthel et al., 2007), donde se plantea la fusión de sensores radar y AIS basándose en el uso de un tracker distribuido DMHT (Distributed Multi Hypothesis Tracker). En (García et al., 2010), también se fusionan AIS y radar siguiendo una arquitectura distribuida, donde cada sensor es encargado de procesar su información localmente. La información de todos los sensores es integrada posteriormente en un nodo central que proporciona la información fusionada a un sistema VTS (Vessel Traffic Service). Este trabajo evolucionó en un nuevo desarrollo que además permitía añadir configuración contextual, que por ejemplo permite cambiar parámetros y algoritmos (Kalman, IMM, PDA, MHT) dependiendo de la zona de operación (Marti et al., 2014).

Si consideramos el uso de la fusión de datos meramente en el entorno de las redes de sensores visuales, podemos encontrar diversos trabajos donde se aplica con buenos resultados. En el estudio propuesto por (Dockstader and Tekalp, 2001) cada cámara realiza un seguimiento independiente de los objetivos que luego es integrado utilizando una red bayesiana de creencias, que fusiona las observaciones de las diferentes cámaras resolviendo relaciones de independencia en el grafo. En el trabajo realizado por (Snidaro et al., 2004, 2003) se utilizó la fusión de datos para mejorar la precisión de las trayectorias de personas detectadas por diferentes cámaras mediante un sistema de fusión centralizado. Se demostró que la fusión de datos reducía el error en la posición que inducía cada cámara por errores en segmentación y calibración. El trabajo propuesto por (Castanedo et al., 2010) es muy similar al anterior en el sentido que se busca mejorar la precisión en la posición, pero lo hace mediante un enfoque distribuido. Esta arquitectura se basa en un sistema multi-agente donde cada nodo procesa localmente la información de la cámara, y luego esta información es transmitida a un agente de fusión que finalmente realiza la tarea.

También en (Wu et al., 2003) se emplea la fusión de datos para mejorar la estimación de la posición de vehículos. Para ello emplean una jerarquía de dos niveles de filtros de Kalman. El nivel más bajo estima posición, velocidad y aceleración en coordenadas locales a cada cámara, mientras que el segundo nivel integra la información de cada cámara en un plano de referencia común, y aplica otro filtro Kalman. También nos podemos encontrar con métodos alternativos para fusionar información de diferente naturaleza, como el vídeo a color y el procedente del

espectro infrarrojo (Conaire et al., 2006), donde ambos sistemas se pueden complementar para satisfacer diferentes condiciones ambientales. Inclusive algunos trabajos proponen el uso de la fusión de datos para mejorar sistemas donde el objetivo es el reconocimiento de actividades, como el propuesto en (Cilla Ugarte, 2012).

### 2.2.3 Modelos de fusión

En el transcurso de la fusión de datos se han desarrollado múltiples modelos de fusión (Esteban et al., 2005; Foo and Ng, 2013) que permiten de una manera conceptual comprender o identificar formas de afrontar el problema. Estos modelos permiten además servir de marco teórico sobre el que diferentes personas pueden entenderse dentro del dominio del problema. Sin embargo en ningún caso se describe cómo se soluciona un problema o los algoritmos a emplear.

Podemos destacar los siguientes modelos de fusión desarrollados durante las últimas décadas:

**Ciclo de Inteligencia**

Este modelo (Bedworth and O'Brien, 2000; Shulsky and Schmitt, 2002) define cuatro fases sobre las que modelar el proceso de fusión de datos. El primer paso se describe como adquisición, que consiste en el despliegue de un conjunto de sensores o fuentes de información humana para obtener información de inteligencia, que normalmente será presentada como un informe de inteligencia (aún de alto nivel). El segundo paso, llamado colación, consistiría en realizar un proceso de análisis, comparación, y correlación de los informes de inteligencia asociados. En el tercero contamos con un proceso de evaluación, cuyo objetivo sería generar información fusionada y analizada de los diferentes informes de inteligencia. El último paso, llamado diseminación, sería la etapa donde la información de inteligencia fusionada se transmite a los usuarios u operadores que estarían encargados del proceso de toma de decisiones.

**Bucle de control Boyd**

También conocido como el bucle Observar, Orientar, Decidir y Actuar (OODA) (Das, 2008; Nilsson, 2007; Plehn, 2000). Fue inicialmente propuesto para modelar los comandos militares. Al igual que el modelo de ciclo de inteligencia, describe cuatro fases. La primera, llamada Observar trataría de adquisición de la información del entorno. Una vez hemos obtenido la información llegaría la fase de orientar, que básicamente consistiría en la evaluación del estado actual y las amenazas. Una vez analizada la información del entorno, llegaría la fase de Decidir, que consiste en tomar las decisiones que se van a llevar a cabo para dar respuesta a la situación del entorno. La última fase ya consistiría en la fase de Actuar, que consiste en ejecutar las tareas planificadas en la fase de decisión. El énfasis puesto en este modelo trata de acortar el tiempo de respuesta entre la fase de observación y actuación. Permitiendo obtener ciertas ventajas en la batalla.

### Modelo JDL

El modelo de fusión JDL se propuso para categorizar las diferentes funciones relacionadas con la fusión en diferentes niveles. Hablaremos en más detalle de este modelo en la sección 2.2.4, ya que este es uno de los modelos de fusión más extendidos y utilizados, y sobre el que nos basaremos para el desarrollo de la arquitectura multi-agente.

### Modelo en cascada

El modelo de fusión en cascada (Esteban et al., 2005; Gad and Farooq, 2002) define un modelo de fusión jerárquico que consta de tres niveles de representación:

- Nivel 1 (monitorización, procesado de señal): Consiste en la correcta transformación de la información en crudo generada por los sensores para su correcto aprovechamiento.
- Nivel 2 (extracción de características, detección de patrones). Con el fin de minimizar el contenido de la información, y maximizar la información transmitida, en esta fase se realizarían los procesos de fusión de información.
- Nivel 3 (evaluación de la situación, toma de decisiones). En este nivel se describirían las posibles vías de actuación dada la información del entorno, las relaciones entre objetos y los eventos, etc.

### Modelo Dasarathy

Los procesos de fusión de información normalmente se han relacionado con un proceso jerárquico con tres niveles generales de abstracción: la información (generada por los sensores), las características (información de sensores más procesada y refinada), y las decisiones de actuación. Sin embargo, Dasarathy (Dasarathy, 1994, 1997) observó que el proceso de fusión puede ocurrir simultáneamente dentro de todos estos niveles. El modelo de Dasarathy fue propuesto para expandir el modelo jerárquico comúnmente utilizado a cinco categorías de fusión basadas en módulos que definen una serie de entradas y salidas. *Data in-data out fusion* trataría de la fusión a nivel de datos. *Data in-feature out fusion* trataría de la selección y extracción de características. *Feature in-feature out fusion* está relacionado con la fusión de información a nivel de características. *Feature in-decision out fusion* tiene que ver con la extracción y procesamiento de patrones que generen un conjunto de decisiones. *Decision in-decision out fusion* trataría de la fusión de decisiones.

### Modelo Visual de Fusión de datos

Este modelo fue propuesto por Karakowski (Bossé et al., 2007; Karakowski, 1998) como una extensión del modelo JDL, donde se consideraba la inclusión integral de un humano en el proceso de fusión. Como los humanos nos guiamos por las percepciones visuales, este modelo se centra en la información visual que el humano debía percibir para solucionar

los problemas o tomar las decisiones. Las premisas que se describen en el modelo VDF son las siguientes:

- El humano es un elemento central que participa activamente en el proceso de fusión, generando soluciones creativas a los problemas.
- La información derivada del proceso de fusión visualizada por el humano se usa principalmente para ayudarle a obtener una percepción global de la situación, a la vez que proporcionarle posibles formas de solucionar los problemas.
- Se deberá proporcionar algún tipo de representación visual con el fin de minimizar la información requerida por el humano para solucionar el problema.

**Modelo Omnibus**

El modelo Omnibus fue propuesto por Bedworth y O'brien (Bedworth and O'Brien, 2000), y en él se pretendía unificar los modelos del Ciclo de Inteligencia, el modelo JDL, el bucle OODA, el modelo de Dasarathy, y el modelo en cascada.

**Modelo Endsley**

El modelo de Endsley (Bossé et al., 2007; Endsley, 1995; Endsley et al., 2000) es utilizado con frecuencia para modelar el estado del entorno. Es un modelo cognitivo que usa una definición general del estado que puede ser aplicado a múltiples dominios: El conocimiento del entorno consiste en la percepción de los elementos dentro de un espacio de tiempo y espacio, la comprensión de su significado, y la proyección de su estado a un futuro próximo. Por lo tanto, este modelo se divide en tres fases jerárquicas que se definen como:

- Nivel 1: Percepción de los elementos del entorno. Como puede ser su estado, los atributos, y la dinámica de los elementos relevantes.
- Nivel 2: Comprensión de la situación actual. Basada en los elementos detectados en el nivel 1. Trataría con la integración de múltiples partes de información y la determinación de su relevancia para satisfacer los objetivos del operador.
- Nivel 3: Proyección del estado. Relacionado con la habilidad para predecir o anticiparse a eventos y dinámicas futuras de los elementos del entorno. Esta se realizaría gracias a la información obtenida en los niveles 1 y 2, y permitiría tomar decisiones a tiempo.

### 2.2.4 Modelo de fusión JDL

El modelo de fusión JDL (Llinas et al., 2004) fue presentado a comienzos de los 90 como un modelo conceptual para servir de ayuda en el desarrollo de sistemas de fusión. El modelo está compuesto por diferentes niveles con diferentes niveles de abstracción. Desde los procesos de

más bajo nivel que pueden estar relacionados con el procesamiento de la información, a los procesos de más alto nivel que puede comprender la evaluación del estado y la gestión de los sensores. El objetivo principal de este modelo es proporcionar un marco teórico sobre el que diferentes grupos de interés se puedan entender cuando hablan de problemas de fusión.

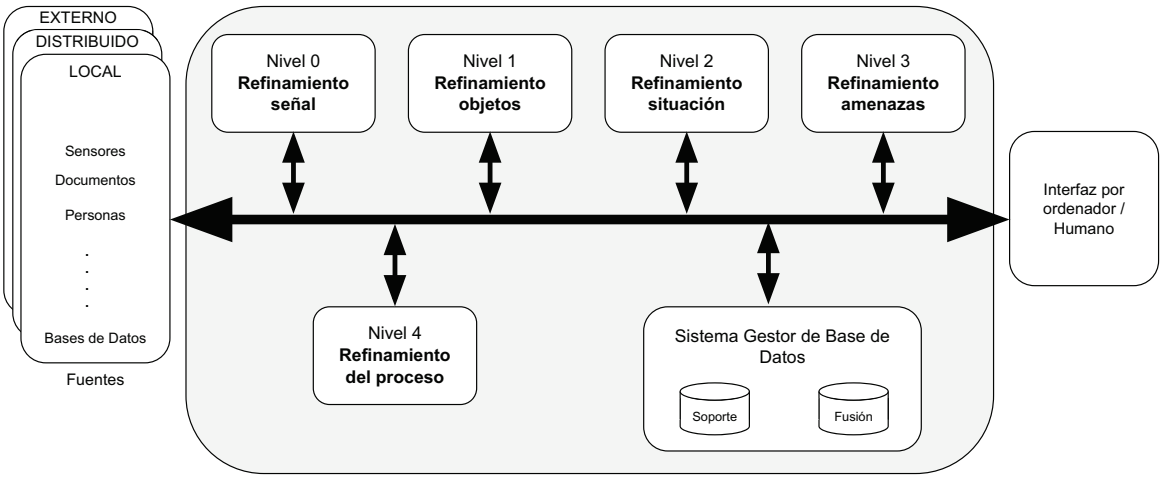


Figura 2.1: Modelo de fusión de datos JDL que contempla diferentes niveles a tener en cuenta en la integración de información multi-sensor.

En este se definen diferentes niveles a contemplar para la integración de información en entornos multi-sensor, tal y como se puede observar en la figura 2.1. Podemos resumir estos brevemente a continuación:

**Nivel 0:** Está relacionado con el refinamiento a nivel de señal. Es decir, el refinamiento de la información proporcionada por el sensor para que pueda ser procesada adecuadamente. Por ejemplo, en un sistema de visión por computador esto podría suponer realizar el proceso de la señal de vídeo para generar un conjunto de regiones de detección dentro de la imagen, como la extracción de blobs.

**Nivel 1:** En este nivel trataríamos de realizar la identificación y seguimiento de los diferentes objetos detectados en el entorno. Por lo que aquí podríamos integrar información de múltiples sensores utilizando algoritmos de alineación temporal, asociación, y correlación, etc. Con la información proporcionada por este nivel podríamos tener una representación global, no redundante, de los diferentes objetos detectados en el entorno.

**Nivel 2:** Una vez contamos con la información de los diferentes objetos, a este nivel se evaluaría la situación del entorno, es decir identificar las relaciones entre objetos o la relación de los objetos con el entorno. Esto nos permitirá hacernos una idea del estado actual del entorno, y por tanto poder tomar decisiones.

**Nivel 3:** Este nivel se buscaría identificar las posibles amenazas. Si en el nivel 2 podemos conseguir evaluar la situación actual del entorno, en este nivel básicamente se evaluarían las posibles amenazas a futuro en el caso de no tomar alguna medida o decisión. Nos permitirá reaccionar y tomar medidas para evitar situaciones no deseadas.

**Nivel 4:** Este trataría de un nivel de refinamiento del proceso completo. Es decir, aquí tendría cabida cualquier proceso de gestión que permita mejorar de algún modo el proceso completo de fusión a cualquiera de los niveles.

Como podemos ver, este modelo representa la mayoría de los problemas a tratar en un en entorno multi-sensor. Contamos con una serie de sensores que procesan y proporcionan información sobre el entorno (Nivel 0). Necesitamos un mecanismo de integración de información que nos permita obtener una representación fiable de las diferentes percepciones multi-sensor (Nivel 1). Como además estamos diseñando una arquitectura de gestión autónoma, será necesario poder evaluar la situación del entorno y las decisiones que es necesario tomar a cabo para satisfacer los objetivos impuestos (Niveles 2 y 3). Y por último realizar el proceso de gestión que permita satisfacer el objetivo, ya sea a través de la gestión del sensor, la activación de un actuador, o el refinamiento de un algoritmo (Nivel 4). Además, no podemos olvidar la interacción con el operador y la información que podremos necesitar presentarle.

Este modelo por sí sólo no impone un orden de ejecución entre los diferentes niveles. De hecho varios de ellos podrían encontrarse ejecutando en paralelo, especialmente en entornos distribuidos donde contemos con información monitorizada por los sensores, y un operador o sistema autónomo analizando la información y realizando tareas de gestión.

## 2.3 Gestión de sensores

### 2.3.1 Introducción

Hemos podido observar mediante el estado del arte de la fusión de datos, que en un entorno multi-sensor será necesario considerar la integración de las diferentes fuentes de información. Sin embargo, también hemos podido comprobar a través del modelo JDL, que la fusión de datos también contempla un nivel de refinamiento del proceso (nivel 4). Además de integrar la información de manera inteligente, la fusión de datos debe incorporar técnicas proactivas y reactivas de planificación y gestión del conjunto de recursos como pueden ser los sensores, con el fin de hacer el mejor uso posible dentro de los requisitos especificados para el entorno.

Por lo tanto, la gestión multi-sensor se puede describir formalmente como el sistema o proceso en el que se busca la gestión o coordinación de un conjunto de sensores, o dispositivos de medida, en un entorno dinámico y con incertidumbre, para mejorar el rendimiento en la fusión de datos y la percepción que recibimos (Xiong and Svensson, 2002). El porqué y qué problemas plantean tanto la gestión de un sensor como la gestión multi-sensor se ha discutido a lo largo de numerosos trabajos, como los que podemos encontrar en (Katsilieris, 2015), (Farmani et al., 2014), (Ng and Ng, 2000), (Blackrnan and House, 1999), (Lopez et al., 1998), y (Musick and Malhotra, 1994). El objetivo básico de la gestión multi-sensor será seleccionar los sensores adecuados para realizar la tarea adecuada en el momento oportuno. Para esto, un sistema de gestión de sensores tendrá que responder a las siguientes preguntas:

¿Qué tareas de observación es necesario realizar y cuáles son sus prioridades?

¿Cuántos sensores se requieren para proporcionar la información necesaria?

¿Qué sensores es necesario desplegar, y en qué posición?

¿Qué conjunto de sensores van a realizar las tareas?

¿Cuál es el modo de funcionamiento o la secuencia de acciones para un sensor particular?

¿Qué valores de parámetros deben ser seleccionados par la operación de los sensores?

La tarea más sencilla de gestión podría consistir en seleccionar el parámetro de sensor óptimo dados un conjunto de sensores con respecto a una tarea particular (Tarabanis et al., 1995). Los problemas de gestión multi-sensor más generales, sin embargo, estarán relacionados con las decisiones de qué sensores usar para cada tarea, además de cuando y dónde usarlos.

Para comprender mejor el problema de la gestión multi-sensor, describiremos en la sección 2.3.2 los diferentes retos que se nos pueden plantear. Comentaremos las arquitecturas de gestión más habituales en la sección 2.3.3. En el punto 2.3.4 se abordarán los enfoques más relevantes para solucionar los problemas de gestión. Y por último, en la sección 2.3.5 comentaremos la gestión distribuida utilizando sistemas multi-agente, debido a la relación que tendrá con la propuesta de esta tesis.

### 2.3.2 Problemas de la gestión multi-sensor

#### 2.3.2.1 Distribución espacial

El despliegue de sensores es un problema crítico relacionado con la captura de información eficiente en un entorno dinámico. Este se centra en tomar decisiones sobre cuándo, dónde, y cómo desplegar los recursos de monitorización en reacción a los cambios de estado producidos en el entorno (Iyengar and Brooks, 2012). El algunos casos, podría ser beneficioso desplegar recursos proactivamente de acuerdo a la predicción de la evolución del entorno con el fin prepararse para observar un evento futuro.

La ubicación de los sensores toma especial atención en el despliegue de los sensores. Este problema consiste en el posicionamiento de múltiples sensores de forma simultánea en ubicaciones óptimas o casi óptimas con el fin de dar soporte a tareas de vigilancia. Lo normal es poder ubicar los sensores en una región particular determinada por situaciones tácticas optimizando determinados criterios, normalmente expresados en términos de probabilidad de detección global, calidad de las pistas, etc. Este problema puede ser formulado como problemas de optimización con restricciones.

Estas restricciones pueden estar determinadas por alguno de los siguientes factores:

- Los sensores estarán normalmente restringidos a regiones específicas debido a consideraciones tácticas, alcance de la conectividad, calidad de servicio, autonomía, etc. (Lee et al., 2009)
- Algunas restricciones pueden estar impuestas en base a la posición relativa de sensores adyacentes con el fin de permitir su comunicación cuando los sensores se encuentran distribuidos en una red descentralizada. O bien cuando hay que tener en cuenta la cobertura de cada sensor para cubrir un área determinada (Ahmed et al., 2005).
- La cantidad de recursos de monitorización que pueden posicionarse en un determinado período de tiempo están limitados por restricciones logísticas o por el tiempo de exposición máximo antes de ser detectado (Zou and Chakrabarty, 2003).

En los casos más sencillos, la decisión de la ubicación de los sensores se pueden tomar en base a la descripción del entorno, que en teoría es estacionario. Como ejemplos, se pueden considerar los siguientes casos:

- Ubicar los sensores en posiciones que minimicen el efecto de apantallamiento del terreno o edificios en la detección de aviones, embarcaciones, monitorización de personas o coches de un determinado lugar (Murray et al., 2007).

- Despliegue de un conjunto de sensores sobre una región específica para obtener la mayor cantidad de información posible (Dhillon and Chakrabarty, 2003).

En estos escenarios se podrían emplear modelos matemáticos o físicos como pueden ser modelos digitales del terreno o de la propagación de la señal en el entorno. Estos modelos podrían ayudar a tomar decisiones a la hora de ubicar los sensores. Sin embargo, podríamos encontrarnos sistemas más complejos donde el entorno es cambiante y puede ser necesario reposicionar repetidamente los sensores para refinar y actualizar el estado de los objetivos que se encuentra en movimiento en tiempo real (Wang et al., 2006). Estos casos se pueden dar en entornos como los siguientes:

- Seguimiento de submarinos a través del despliegue de sonoboyas pasivas en misiones de lucha antisubmarina (Johansson et al., 1997).
- Localización de objetivos móviles usando receptores de emisiones electrónicas (ESM) (Spezio, 2002).
- Seguimiento de tanques en tierra desplegando sensores acústicos pasivos (Becker and Güdesen, 2000).

#### 2.3.2.2 Asignación de tareas

El propósito básico de la gestión de sensores es adaptar el comportamiento del sensor a entornos dinámicos. Por asignación de comportamiento nos referimos a la determinación eficiente en la planificación de las funciones del sensor y su utilización de acuerdo a la situación cambiante, teniendo en cuenta los requisitos de la tarea de vigilancia o monitorización (Xiong and Svensson, 2002). En este caso aparecen dos puntos importantes a tratar:

- Decidir el conjunto de tares de observación, a nivel global, que los sensores deberían realizar teniendo en cuenta el estado actual o el futuro próximo, en base a la evaluación o predicción del estado del entorno para satisfacer el objetivo global de monitorización.
- Planificar y programar las acciones a realizar por cada uno de los sensores desplegados para atender las tareas de observación propuestas y sus objetivos.

Teniendo en cuenta el entorno real y los recursos de monitorización limitados, es entendible que no se puedan satisfacer simultáneamente todas las tareas de monitorización deseadas. De manera intuitiva, las tareas más urgentes o importantes deberían obtener mayor prioridad en su competición por la asignación de recursos. Por lo que es necesario un modo de priorizar las tareas de observación. Esta información puede ser muy útil en la planificación de las acciones de los sensores y la negociación entre sensores en un entorno descentralizado.

Por ejemplo, podemos considerar un escenario de vigilancia que incluya un conjunto de objetivos de monitorización a través de un conjunto de sensores que pueden centrar la atención en los diferentes objetivos con diferentes modos de seguimiento y/o clasificación. El primer paso en la gestión del sensor será utilizar las evidencias adquiridas del entorno para decidir los objetivos de interés y priorizar los que vamos a monitorizar. De esta forma, en el siguiente paso, podríamos ajustar los diferente sensores y sus modos de funcionamiento para que se encuentren en mejor disposición para obtener información de los objetivos.

Teniendo en cuenta las restricciones en recursos y el procesamiento de la información, en general no es posible monitorizar todos los objetivos de interés con todos los sensores en un intervalo de tiempo dato. Además, la mejora de precisión de monitorización en un objetivo puede generar una degradación de rendimiento sobre la monitorización de otro objetivo. Por lo que será necesario buscar un compromiso aceptable entre los diferentes objetivos.

Podemos encontrar numerosos trabajos relacionados con este tipo de problemática de gestión multi-sensor. A menudo aparecen bajo diferentes términos, como planificación de sensores (Kristensen, 1996), selección de sensores (Giraud and Jouvencel, 1994; Kalandros and Pao, 1998; Kalandros et al., 1999), y también asignación de sensores a tareas (Lopez et al., 1998; Molina Lopez et al., 1995). Todos estos términos significan básicamente lo mismo, la distribución de recursos entre un conjunto de tareas de observación.

#### 2.3.2.3 Coordinación de sensores en una red descentralizada

Hay dos formas generales de integrar un conjunto de sensores dentro de una red de sensores. Una de ellas es la arquitectura centralizada, donde todas las acciones de todos los sensores se deciden por un nodo central del sistema. Este tipo de arquitecturas son más fáciles de desarrollar, ya que se cuenta con toda la información del entorno en un único nodo, sobre los que es posible aplicar algoritmos de fusión, planificación, optimización, asignación de tareas, etc. Sin embargo estas arquitecturas pueden sufrir de problemas de saturación, baja tolerancia a fallos, poca escalabilidad, etc.

La otra alternativa puede consistir en tratar a los sensores de la red como agentes inteligentes distribuidos que cuentan con un determinado grado de autonomía (Wesson et al., 1981). No haría falta contar con un único nodo de gestión, si no que podríamos tener múltiples agentes inteligentes para comunicarse y coordinarse sobre las diferentes tareas a realizar. Esto permitiría solucionar muchos de los problemas de una arquitectura centralizada, pero complica el diseño a nivel de planificación de tareas, coordinación, etc. Por lo que este sería otro de los problemas de la gestión multi-sensor cuando pensamos en la integración a través de una red descentralizada.

Podemos encontrar diferentes trabajos relacionados con la gestión descentralizada de sensores y lo problemas que estos plantean, desde problemas de fusión distribuidos (Hall et al., 2012; Kumar et al., 2003), a la gestión distribuida para optimizar el seguimiento de múltiples

objetivos (Fu et al., 2012; Ling et al., 2011), a la utilización de procesos de decisión de Markov jerárquicos (Akselrod et al., 2007), y otra serie de trabajos relacionados (Bevington, 2004; Chen et al., 2005; Mayott et al., 2010; Shen et al., 2011).

### 2.3.3 Arquitecturas de gestión

La arquitecturas de gestión que se pueden desarrollar sobre un sistema multi-sensor se diferencian básicamente en la capacidad de gestión interna del sensor. En general, se pueden definir los sistemas centralizados como aquellos en los que las decisiones que afectan a todo el sistema se toman en un único nodo y los descentralizados como aquellos donde estas se toman entre todos los nodos sin que exista una supeditación de unos nodos a otros.

A medida que se permite a los sensores una mayor capacidad de autonomía y por lo tanto se les dota de una mayor capacidad para la toma de decisiones, resulta menos necesario centralizar la toma de decisiones del sistema en único punto. Esta autonomía redunda en la arquitectura de gestión al permitir que las decisiones se tomen mediante la puesta en común de decisiones locales.

Dentro de las arquitecturas que podemos encontrar hoy día se encuentran las siguientes:

**Gestión Centralizada**

En los sistemas centralizados uno de los nodos es el líder del grupo, las decisiones fluyen desde él al resto de nodos y son aceptadas en el proceso de decisión de cada nodo. Este es el caso de redes multi-sensor donde los sensores tienen una capacidad limitada en la toma de decisiones. Esta limitación conduce a la existencia de un nodo que toma las decisiones que afectan a todo el sistema, el centro de fusión, que debe realizar asumir la tarea de integración de información de múltiples sensores, evaluar la importancia de las tareas de vigilancia del sistema, y la posterior asignación de tareas a cada uno de los sensores, en función de la capacidad y la importancia que representa cada tarea para cada sensor. En este caso, el centro de fusión informa a cada uno de los sensores de las tareas que debe realizar, asignándoles un grado de importancia que cada sensor aplicará a la hora de ejecutar las tareas.

**Gestión Distribuida**

En un sistema descentralizado la información es fusionada localmente con un conjunto de agentes locales en vez de una unidad central. En este caso, cada sensor o plataforma de sensores pueden ser vistos como un conjunto inteligente que tiene cierto grado de autonomía en la toma de decisiones. La coordinación de los sensores es conseguida a través de la comunicación de la red de agentes, en donde cada sensor comparte localmente información fusionada y cooperan entre ellos (Durrant-Whyte et al., 2001).

Las ventajas que aportan este tipo de arquitecturas son las siguientes:

- Arquitectura escalable, ya que no se encuentra limitada por la capacidad de procesamiento o ancho de banda de un nodo centralizado que realiza todas las tareas.
- Más tolerante a fallos, ya que si se cae un nodo de la arquitectura, el resto de nodos puede seguir funcionando. En una arquitectura centralizada, la pérdida del nodo central de fusión podría suponer la pérdida de todo el sistema.
- Modular en el diseño e implementación de los nodos de fusión.

**Gestión Jerárquica**

Esta puede ser vista como una mezcla entre las arquitecturas centralizadas y descentralizadas. En un sistema jerárquico suele haber varios niveles de la jerarquía donde el nodo superior funciona como el centro de fusión global, mientras que los niveles inferiores pueden consistir en múltiples centros de fusión locales (Ng and Ng, 2000). Cada nodo local de fusión es responsable de la gestión de un subconjunto de sensores. La división de todo el conjunto de sensores en diferentes grupos puede ser realizado en función de la localización geográfica de los sensores o las plataformas.

### 2.3.4 Enfoques para la asignación de recursos

Dentro de los problemas de la gestión multi-sensor que hemos descrito, esta tesis se va a centrar principalmente en la gestión de sensores desde el punto de vista de la asignación de recursos. En este problema, dado un conjunto de tareas de observación que se establecen en el sistema, el gestor de los sensores debería ser capaz de distribuir los recursos de monitorización entre los recursos disponibles con el fin de llevar a cabo las tareas en la medida de lo posible. Por lo tanto, en esta sección vamos a revisar diferentes enfoques que se pueden contemplar para resolver este tipo problemas.

#### 2.3.4.1 Teoría de la información aplicada a la gestión de sensores

La teoría de la información está relacionada con las leyes matemáticas que rigen la transmisión y el procesamiento de la información y se ocupa de la medición de la información y de la representación de la misma, así como también de la capacidad de los sistemas de comunicación para transmitir y procesar información.

La idea de usar teoría de la información en la gestión de sensores fue propuesta por primera vez en (Hintz and McVey, 1991). Desde el punto de vista de la teoría de la información, los sensores son aplicados para observar el entorno con el fin de incrementar la información (o reducir la incertidumbre) sobre el estado del entorno. Por lo tanto, la tarea de gestión consistirá en la asignación de los recursos de monitorización a la diferentes tareas de tal forma que se maximice la cantidad de información percibida en cada momento.

En los entornos de vigilancia, la información percibida se referirá a la detección de los diferentes objetivos, así como al incremento de su precisión, o reducción de la incertidumbre. Por ejemplo, la observación de los objetivos por diferentes sensores puede ayudar a reducir la incertidumbre en la posición de una medida de posición por cada actualización recibida. Para cuantificar la información obtenida sobre un objetivo, dada la observación de un sensor, se suelen utilizar dos medidas, principalmente basadas en la entropía de Shannon (Lin, 1991) y la divergencia de Kullback-Leibler (Mahler, 2003; Manyika and Durrant-Whyte, 1994)

En la literatura ambas medidas de información han mostrado su utilidad en la selección de sensores en un entorno multi-sensor y seguimiento multi-objetivo. La entropía de Shannon fue utilizada en (McIntyre and Hintz, 1996; Schmaedeke, 1993), donde la ganancia de información esperada se puede calcular tras la actualización del estado en un filtro de Kalman, observando para ello la covarianza del estado. Por otro lado, en (Schmaedeke and Kastella, 1998) y (Dodin et al., 2000) se aplicó la divergencia de Kullback-Leibler para medir la discriminación de la información utilizando un filtro IMM (Interacting Multiple Model Kalman Filter).

Podemos encontrar muchos más trabajos relacionados con la gestión de sensores utilizando teoría de la información en trabajos como (Aoki et al., 2011; Katsilieris et al., 2012; Kreucher et al., 2007; Zhao et al., 2003, 2002). E inclusive el uso de otras medidas de entropía de información como la entropía de Rényi o también conocida como $\alpha$-divergence (Hero et al., 2007), en trabajos como (Kreucher et al., 2003).

#### 2.3.4.2 Teoría de la decisión aplicada a la planificación de sensores

La teoría de la decisión se ocupa de analizar cómo elige una persona aquella acción que, de entre un conjunto de acciones posibles, le conduce al mejor resultado dadas sus preferencias. El paradigma canónico de la teoría de la decisión se caracteriza por contar con un individuo que ha de tomar una decisión y de quien se dan por supuestas sus preferencias. Así la teoría de la decisión no entra a considerar la naturaleza de las preferencias de los individuos, ni por qué éstos prefieren unas cosas en vez de otras. Lo único que importa es que dichas preferencias satisfagan ciertos criterios básicos de consistencia lógica.

En (Kristensen, 1996) se trató el problema de la selección de tareas de monitorización, dentro de un conjunto de candidatos, como un problema de toma de decisiones, y se desarrolló una arquitectura basada en el Análisis Bayesiano de Decisión (BDA) para su propuesta. Se determinó en este trabajo, que la incertidumbre tiene que ser tratada en la planificación del sensor dado que el último propósito de realizar tareas de monitorización es para reducir la incertidumbre acerca del entorno. Por otro lado, el BDA ofrece un mecanismo potente para representar y razonar bajo incertidumbre. La teoría BDA se utilizó en principio en el área de la economía para evaluar diferentes acciones, como por ejemplo evaluar diferentes inversiones.

Similar al trabajo propuesto por (Kristensen, 1996), fue el trabajo presentado por (Lindner et al., 1994). En este se proponía un método que estimaba la utilidad esperable de cada sensor en su uso dentro de un sistema de fusión. Los valores de utilidad calculados eran entonces utilizados para predecir el subconjunto de sensores a usar en la monitorización del entorno con el fin de minimizar el coste de cada ciclo de observación.

#### 2.3.4.3 Gestión de recursos usando lógica borrosa

La lógica borrosa es una especialización de la inteligencia artificial que se basa en el concepto "Todo es cuestión de grado". Esto permite manejar información vaga o de difícil especificación en sistemas que requieren su evaluación para cambiar el estado en un sistema específico. Es entonces posible con la lógica borrosa gobernar un sistema por medio de reglas de sentido común, las cuales se refieren a cantidades indefinidas.

Las reglas involucradas en un sistema borroso, pueden ser aprendidas con sistemas adaptativos que aprenden al observar como operan las personas o los dispositivos reales. Aunque estas reglas pueden también ser formuladas por un experto humano. En general la lógica borrosa se aplica tanto a sistemas de control como para modelar cualquier sistema continuo de ingeniería, física, biología o economía.

La lógica borrosa es entonces definida como un sistema matemático que modela funciones no lineales, que convierte unas entradas en salidas acordes con los planteamientos lógicos que usan el razonamiento aproximado. Se fundamenta en los denominados conjuntos borrosos y un sistema de inferencia borroso basado en reglas de la forma "si ... entonces ...", donde los valores lingüísticos de la premisa y el consecuente están definidos por conjuntos borrosos.

En (Gonsalves and Rinkus, 1998) se describió un proceso de fusión y gestión de recursos inteligente cuyo nivel 4 del modelo JDL trata de un gestor de recursos basado en lógica borrosa, que es el responsable de evaluar la situación actual del entorno y la información de los objetivos enemigos, para generar un conjunto de tareas de monitorización sobre los sensores. La asignación de tareas depende del conocimiento previo de las capacidades de cada sensor y de la táctica llevada a cabo por el enemigo.

Otro trabajo similar se presentó en (Smith III and Rhyne II, 1999) donde se trabaja con la asignación óptima de recursos distribuidos sobre diferentes plataformas de monitorización. Plataformas como barcos o aviones. Los recursos cuentan con varios tipos de sensores como ESM, RADAR, IFF y comunicaciones. Para ello se construyó un árbol de decisión borroso analizando e incorporando información de expertos en el ámbito militar. Como el árbol de decisión es borroso, la incertidumbre en los conceptos de la raíz se propagan cuando se explora el árbol, por lo que se utilizaron técnicas de optimización genética para seleccionar determinados parámetros en los conceptos de la raíz.

Más adelante se propone otro trabajo similar a (Smith III and Rhyne II, 1999) por los mismos autores en (Smith et al., 2000). En este caso se extiende la misma idea pero incorporando técnicas de minería de datos para la optimización del rendimiento del árbol de decisión borroso. En particular, se aplicó un algoritmo genético en el proceso de minería de datos para mejorar el ajuste de las funciones miembro de los conceptos raíz en función de escenarios almacenados en una base de datos.

Podemos encontrar otros trabajos alejados de la gestión de sensores tal y como se conoce dentro del dominio de la fusión de datos que también utilizan lógica borrosa, como el trabajo propuesto en (Xia et al., 2007). En este se trabaja con redes de sensores y actuadores inalámbricos (WSANs) con recursos limitados que necesitan proporcionar una determinada calidad de servicio (QoS). En este caso, cada sensor incorpora técnicas de lógica borrosa en cada sensor para adaptar la frecuencia de monitorización en función de las necesidades dinámicas del entorno.

#### 2.3.4.4 Procesos de decisión de Markov

Un proceso de decisión de Markov (MDP) es un problema de aprendizaje por refuerzo que cumple la propiedad markoviana, es decir, que el estado futuro del proceso es independiente de los estados pasados y solamente depende del estado presente. Viene definido por un conjunto de estados, un conjunto de acciones posibles, un modelo del entorno (opcional) y una función de recompensas, que sirve como medio de interacción entre el entorno y el agente.

El agente o tomador de decisiones puede influenciar el estado del sistema realizando una secuencia de acciones que logran que el sistema optimice su desempeño de acuerdo a algún criterio. El agente observa el estado del sistema en momentos específicos y reúne la información necesaria para tomar acciones, donde cada acción realizada por el agente tiene un costo o una recompensa. Las acciones afectan el estado del sistema e influyen entonces en decisiones futuras.

Este tipo de procesos de decisión, y en concreto los Procesos de Decisión de Markov Parcialmente Observable (POMDP) (Dutech and Scherrer, 2013) han sido utilizados para tareas de la gestión como la monitorización adaptativa, la clasificación de objetos, y el seguimiento adaptativo (Chong et al., 2009). En este tipo de entornos podemos considerar el problema de la gestión de sensores como un problema que trata de encontrar una secuencia de decisiones que minimice el coste de la tarea que está sujeto a una serie de restricciones.

Por ejemplo, en el problema de la clasificación de objetos, se parte de un objeto que no está clasificado y de un sensor que podemos controlar. Cada vez que realizamos una acción de control sobre el sensor, obtenemos una medida sobre el objeto que nos podría permitir su clasificación, pero esta tendría un coste asociado. Después de cada medida, se decide si se puede clasificar el objeto. Si el objeto es clasificado el problema termina, y si no, se puede continuar tomando medidas sobre el objeto. Por lo que en este tipo de problemas interesa poder

maximizar la probabilidad de clasificación teniendo el cuenta el coste total de la operación. Este tipo de problemas se puede modelar mediante POMDP y ser solucionados mediante técnicas de programación dinámica (SPD). Como por ejemplo el trabajo presentado en (Bertsekas et al., 1995).

#### 2.3.4.5 Modelado de la selección de sensores como un problema de búsqueda

La selección de un subconjunto de sensores para aplicar una tarea de monitorización determinada puede ser considerado como un problema de búsqueda en un espacio combinatorio. El objetivo es encontrar la solución más apropiada entre todos los sensores posibles. Para poder llevar a cabo este tipo de problemas será clave la evaluación de cada alternativa dentro del dominio del problema.

Para evaluar la idoneidad de cada sensor, en (Giraud and Jouvencel, 1994) se utilizó teoría de lógica borrosa para crear un grafo de preferencia de sensores, que puede ser explorado para encontrar los subconjuntos de sensores para realizar que permiten realizar una tarea determinada. Molina (Molina Lopez et al., 1995) utilizó por su parte una serie de heurísticas asociadas a cada nodo del árbol de búsqueda para la evaluación de un subconjunto de sensores en términos de la carga del sensor y su idoneidad. La carga del sensor constituye una estimación de los recursos del sensor requeridos para llevar a cabo la tarea, mientras que la idoneidad del sensor para realizar la tarea se define como una función sobre la necesidad de completar la tarea y la habilidad del sensor para realizarla. En este caso, la función objetivo para el algoritmo de búsqueda debe ser construida de manera adecuada para reflejar un buen balance entre los dos factores.

En (Kincaid and Laba, 1998) por otro lado se examinó la Búsqueda Tabú Reactiva (RTS) para ser aplicada en el problema de selección de sensores en el control activo acústico estructural. El objetivo de este problema consistía en seleccionar un conjunto de 8 sensores sobre 462 posibles ubicaciones de tal forma que el ruido medido con los 8 sensores elegidos sea muy similar al ruido que podrían medir los 462 sensores.

En (Perera et al., 2013) por otro lado se centran en la cantidad de sensores que son desplegados hoy día gracias al Internet de las Cosas (IoT), y la necesidad de poder buscar y seleccionar sensores utilizando información de contexto y prioridades del usuario. Para ello tienen en cuenta múltiples características como la fiabilidad de los sensores, su precisión, tiempo de vida de la batería, etc. Para evaluar la adecuación de los sensores se utilizan distancias euclídeas ponderadas con las prioridades del usuario.

#### 2.3.4.6 Heurísticas

Las heurísticas se refieren a un conjunto de reglas (o ajustes a medida diferentes a un resultado óptimo) que dictan el comportamiento del sensor o el gestor de recursos. Durante muchos años, las heurísticas han sido el caballo de batalla de la gestión de sensores, tal y como se describe en (Blackrnan and House, 1999).

Este conjunto de reglas se crea y se ajusta hasta que el sensor muestra un comportamiento o un rendimiento esperado. Estas reglas dependen específicamente del contexto operacional y las necesidades de operación del propio sensor.

Las reglas también pueden ser definidas en base a la observación de funciones objetivo. Estas reglas tratan de imitar el comportamiento de una función objetivo seleccionado la tarea de monitorización a realizar pero a un coste computacional mucho menor que realizando la optimización de la función objetivo (Charlish et al., 2012).

Por ejemplo, en (Joshi and Boyd, 2009) se considera el problema de seleccionar un conjunto mínimo de medidas (k) sobre un amplio conjunto de posibles sensores (m) de tal forma que se minimice el error en la estimación de determinados parámetros. Solucionar este problema evaluando cada una de las posibles combinaciones (m k) no es práctico salvo que el número de m y k sean pequeños. En este caso se propone una heurística basada en optimización convexa que resuelve de manera aproximada el problema. Esto les permite reducir la complejidad del problema para cualquier número de m y k, obteniendo un buen compromiso en términos de rendimiento y estimación.

### 2.3.5 Gestión distribuida mediante sistemas multi-agente

En entornos descentralizados podemos pensar en sistemas multi-sensor donde los sensores son considerados como miembros o agentes de un equipo, capaces de razonar y tomar decisiones en grupo. En este caso, el concepto de equipo o comunidad se refiere a una asociación flexible en vez de una conexión rígida de sensores. Para este tipo de tareas se podrían considerar algunos principios del comportamiento social e integrarlo en una red de sensores para facilitar la cooperación entre miembros individuales y resolver los objetivos de forma conjunta. Este tipo de consideraciones puede ser integrado fácilmente con tecnología basada en agentes que gestionen los requisitos temporales e implementen los comportamientos sociales dentro de una comunidad de sensores (Xiong and Svensson, 2002).

Un agente es un sistema informático, situado en algún entorno, dentro del cual actúa de forma autónoma y flexible para así cumplir sus objetivos. Además de la interacción con el medio, un agente se caracteriza, utilizando la definición de (Wooldridge and Jennings, 1995), por las siguientes propiedades:

- Autonomía: tiene la capacidad de actuar sin intervención humana directa o de otros agentes.
- Sociabilidad: capacidad de interaccionar con otros agentes, utilizando como medio algún lenguaje de comunicación entre agentes.
- Reactividad: un agente está inmerso en un determinado entorno (hábitat), del que percibe estímulos y ante los que debe reaccionar en un tiempo preestablecido.
- Iniciativa: un agente no sólo debe reaccionar a los cambios que se produzcan en su entorno, sino que ha de tener un carácter emprendedor y tomar la iniciativa para actuar guiado por los objetivos que debe satisfacer.

Para algunos investigadores el término agente tiene una caracterización más concreta, para que un sistema informático pueda considerarse como un agente tiene que modelarse e implementarse usando aspectos que usualmente se aplican a los humanos. Así, en (Shoham, 1993), se define un agente como una entidad cuyo estado está formado por componentes mentales (típicas de los humanos), como creencias, capacidades, elecciones y compromisos: estados que determinan las acciones que llevan a cabo los agentes y que están afectados por los mensajes que reciben.

En la literatura podemos encontrar algunos trabajos relacionados con la aplicación de los sistemas multi-agente a la gestión de sensores. Uno de ellos es el trabajo realizado por Molina (Molina et al., 2004), donde se trabaja en entornos distribuidos de vigilancia como los radares para defensa aérea. El objetivo global de la red es monitorizar y seguir todos los objetos que pasan dentro del área cubierta, que será el resultado de la unión de las coberturas de los sensores individuales. Estas tareas pueden realizarse si los nodos en la red cooperan intercambiando información acerca de los objetos móviles que se mueven en las zonas solapadas, de manera que se puede hacer una asignación óptima de recursos de detección y procesado local en los sensores para cubrir los objetivos de alto nivel (vigilancia de toda la cobertura con un retardo mínimo de actualización, seguimiento de todos los objetos de interés en el escenario, con tasas de refresco en función de su prioridad, etc.). La aplicación pionera de este tipo fue DVMT (Distributed Vehicle Monitoring Testbed) (Lesser and Corkill, 1989), que permitió además poner a prueba muchos de los conceptos del desarrollo de las técnicas multi-agente.

También podemos encontrar esta filosofía de gestión distribuida de sensores al campo de la visión artificial con cámaras activas. En este caso, el objetivo es automatizar una gestión coordinada de la red de cámaras desplegadas en un cierto espacio de vigilancia, determinando la asignación espacio temporal de las cámaras (qué zonas son cubiertas en cada momento) así como el procesado de los datos (qué objetivos son seguidos por cada nodo). Podemos destacar el proyecto de Visión Artificial Distribuida desarrollado en la universidad de Tokio (Ukita and Matsuyama, 2002).

Éste es un proyecto desarrollado desde 1996 en la Universidad de Tokio, con la financiación del programa de investigación de la sociedad japonesa de promoción de la ciencia. El objetivo básico es la interconexión de estaciones de observación (procesadores en tiempo real de imágenes con cámaras activas) y robots móviles con visión, para llevar a cabo una comprensión robusta y flexible de la escena y asignación apropiada de recursos según las condiciones cambiantes del escenario. El principal interés del proyecto es la integración de percepción, acción y comunicación para formar sistemas inteligentes. Se propone un sistema de coordinación de cámaras activas para detección y seguimiento de intrusos, utilizando agentes inteligentes asociados a cada cámara.

Cada elemento del sistema es un agente inteligente de visión artificial, el agente cámara (AVA, Active Vision Agent), que está compuesto por dos módulos básicos que se distinguen por su finalidad: percepción y ejecución de acciones/comunicaciones. El módulo de percepción analiza los datos presentes en las imágenes captadas por la cámara, y el resultado de este análisis se lo comunica al módulo de ejecución para que éste concluya la acción que corresponda. Ambos modos funcionan de forma asíncrona, cada uno de ellos activado con sus propios intervalos de tiempo. En paralelo con el procesamiento de bajo nivel que extrae la información procesada a partir de las imágenes, el agente podrá ejecutar un cierto número de ciclos básicos de razonamiento que son suficientes para concluir una acción final a tomar en consecuencia de la situación en la que se encuentra (ciclo de ejecución).

## 2.4 Gestión de sensores de visón

Se considerará la gestión de sensores de visión como un elemento especial dentro de la gestión multi-sensor convencional. En este caso, un sensor de visión es un sensor que requiere de un mayor procesamiento de la señal, la aplicación de algoritmos de visión, la distribución de la señal, etc. Esto lo convierte en un sensor un tanto particular que merece la pena estudiar de manera independiente, por lo que en esta sección proporcionaremos un estado del arte relacionado únicamente con la gestión de sensores de visión, ya que se utilizarán dentro de las dos aplicaciones propuestas en esta tesis.

### 2.4.1 Introducción

El uso de cámaras en los entornos de vigilancia es probablemente uno de los elementos de seguridad y disuasión más empleados durante las últimas décadas. Dentro del ámbito de la investigación, el uso de cámaras ha introducido grandes retos que aún hoy día no tienen solución. Alrededor de 1985 se comenzó a hacer uso de múltiples cámaras, dónde las investigaciones se centraron en mejorar la precisión de los algoritmos de visión por computador mediante la fusión de datos con múltiples cámaras y sensores.

Por ejemplo, en (Nandhakumar and Aggarwal, 1988) se usaron imágenes normales con imágenes térmicas para mejorar la detección y clasificación de objetos. Más tarde, en los años 90, debido principalmente al reducido coste, el uso de cámaras y sensores se incrementó significativamente para cubrir mayores áreas, seguir objetos, y afrontar mejor el problema de la oclusión. La mayoría de los trabajos en esta década fusionaba la información de múltiples cámaras en un servidor central y trataban de resolver la identificación de objetivos comparando las características de los objetos detectados desde los diferentes puntos de vista de las cámaras. Con esto intentaban obtener un seguimiento más consistente y preciso, pero a costa de obtener y procesar más información redundante del entorno.

Hasta el año 2000, la mayoría de los trabajos en detección y seguimiento de objetos se basaron en el uso de cámaras fijas, que lamentablemente no solían capturar imágenes en alta resolución de los objetivos, especialmente cuando estos se encontraban lejos. Posteriormente, con la llegada de las cámaras PTZ, capturar imágenes de mayor calidad de los objetivos se convirtió en un trabajo más fácil. Inicialmente las cámaras PTZ se controlaban manualmente para enfocar con mayor resolución una región de interés en concreto. Más tarde, las cámaras PTZ se unieron a cámaras fijas y cámaras omnidireccionales en configuraciones de maestro-esclavo. Basándose en las observaciones detectadas por la cámaras fijas, las cámaras PTZ se controlaban automáticamente para enfocar las regiones de interés, como caras de personas, matrículas de coches, etc.

Durante la primera década del año 2000, los investigadores también exploraron arquitecturas basadas en sistemas multi-agente para los sistemas de vigilancia. Más recientemente se ha extendido la coordinación maestro/esclavo por sistemas cooperativos de detección y seguimiento. Se han introducido muchos enfoques formales basados en la teoría de juegos, teoría de decisiones, teoría de control, y otros enfoques que podrían considerarse heurísticos. La introducción de "smart cameras" (cámaras inteligentes) en el mercado también jugó un papel importante en el desarrollo de cámaras con algunas capacidades autónomas. Por lo tanto, la mayoría del trabajo durante los últimos años se ha enfocado a desarrollar sistemas que permitan el control y coordinación de cámaras mediante sistemas autónomos y distribuidos.

A continuación se describirán algunos de los métodos que involucran la gestión de los sensores de visión, desde el procesamiento de la señal, la detección y seguimiento de objetos, a las arquitecturas de gestión comúnmente utilizadas en este tipo de entornos.

### 2.4.2 Adquisición de información del entorno

#### 2.4.2.1 Captura y análisis de vídeo

El uso de sensores de visión dentro del campo de la informática ha constituido y aún hoy día sigue constituyendo uno de los mayores retos a los que nos podemos enfrentar. La única información que proporciona este tipo de sensores consiste en una cuadrícula de colores sin ningún tipo de información adicional. Detectar y reconocer objetos en este tipo de información digital es vital para poder dar soporte a servicios de más alto nivel (Valera and Velastin, 2005)

La disciplina encargada de esta tarea se conoce como visión artificial, o también conocida como visión por computador (del inglés computer vision) (Forsyth and Ponce, 2002). Este término abarca numerosas técnicas que se suelen enfocar a tareas concretas como reconocer matrículas, detectar piezas defectuosas en procesos de calidad, reconocimiento facial, sistemas de vigilancia, imágenes médicas, reconocimiento de caracteres, reconocimiento de huellas dactilares, y otras muchas tareas que se utilizan día a día (Szeliski, 2010). Como la cantidad de técnicas de visión artificial es tan amplia y completamente dependiente del objetivo, en esta tesis se abarcarán principalmente las técnicas relacionadas con el procesamiento de imágenes para la detección de elementos de interés.

Normalmente, en el ámbito de la seguridad con cámaras, interesa detectar personas u objetos con el fin de identificar sus movimientos y actividades. Por ejemplo, seria interesante poder detectar sólo a personas, y por cada persona detectar que actividad está realizando. También poder seguirlas a lo largo del tiempo y saber por donde se han movido, si ha entrado en alguna zona restringida, si se ha acercado a alguna zona restringida, etc. (Collins et al., 2000). Para comenzar con esta tarea, el paso básico consistiría en poder detectar las personas y objetos dentro de la imagen.

#### 2.4.2.2 Detección de objetos

Este proceso es comúnmente conocido como segmentación (Foresti et al., 2005), donde se intentan diferenciar (segmentar) los elementos de interés en cada imagen. El método que quizá haya tenido un mayor impacto en la visión por computador es el algoritmo conocido como background subtraction (Jain and Nagel, 1979; Piccardi, 2004), donde se parte de una imagen de fondo de la escena a supervisar, y sobre las que se buscan las diferencias con la imagen actual. Esta diferencia de imágenes da lugar a una serie de zonas en la imagen que pertenecen a elementos que han sido alterados en la escena, y que probablemente correspondan a personas, objetos, o vehículos. Con esta información sería posible empezar a identificar los objetos detectados y actuar dependiendo de las necesidades del sistema.

Sin embargo, con el uso de cámaras PTZ que permiten cambiar el campo de visión, tenemos que tener en cuenta que no es posible tener una imagen estática como fondo. En este caso sería necesario encontrar métodos específicos que satisfagan esta restricción. Son varios los trabajos que intentan solucionar este problema en concreto, los cuales se describen brevemente a continuación.

**Basadas en flujo óptico:** En el trabajo presentado en (Shibata et al., 2008) se presentaba una novedosa técnica basada en la distorsión del flujo óptico para detectar objetos en movimiento dentro de una escena, inclusive cuando se cambia el campo de visión de la cámara.

**Basadas en fondo adaptativo:** Siguiendo con la idea del uso de una imagen de fondo para cámaras estáticas, en (Collins et al., 2000; Lim et al., 2003) se propone el uso de un algoritmo que adapta automáticamente la imagen de fondo para poder reconocer los objetos que están en primer plano.

**Basadas en selección manual:** No sólo es útil detectar automáticamente los elementos de interés en una imagen. A veces es útil y necesario el poder establecer un elemento en concreto y poder seguirlo a lo largo del tiempo. Por ejemplo, el trabajo de (Everts et al., 2007) utiliza el algoritmo mean shift (Meer, 2003) para seguir objetos establecidos en cámaras PTZ.

**Detección de prominencias:** Del término en inglés Saliency detection, son algoritmos de procesamiento de imagen que permiten detectar objetos predominantes o que se salen de lo habitual en la imagen (El-Barkouky et al., 2012). Estos pueden ser fácilmente aplicables a cámaras PTZ puesto que no dependen de ningún estado estable o pasado.

**Detección de patrones:** Puede ser útil en algunas circunstancias evitar la detección de objetos arbitrarios y centrase en patrones (formas) bien conocidos y de interés. En este aspecto

hay varios trabajos que se centran en reconocer y seguir caras (Comaniciu and Ramesh, 2000; Petrov et al., 2008) a lo largo del tiempo con cámaras PTZ.

**Detección de colores:** A parte de las formas, sería posible hacer un seguimiento de objetos basados en su color. Esta técnica resultaría la más fácil de todas las descritas anteriormente, ya que segmentar una imagen por color es una tarea relativamente sencilla y bien conocida (Deng and Manjunath, 2001; Ozyildiz et al., 2002).

**Detección de contornos:** En determinados entornos puede ser útil detectar objetos basándose en un contorno predeterminado que sirva como base para clasificar los objetos presentes en la escena (Shotton et al., 2005).

#### 2.4.2.3 Calibración de las cámaras

Una vez procesada una imagen y detectado un elemento de interés tras la aplicación de algún algoritmo como los descritos anteriormente, podría ser necesario relacionar la posición de un objeto de la imagen con coordenadas de una posición del mundo real. Esta tarea se suele llevar a cabo a través del proceso de calibración. Este proceso permite determinar los parámetros de la cámara, que nos facilita llevar a cabo esta correspondencia.

Hoy día podemos encontrar una gran cantidad de algoritmos y métodos automáticos de calibración para cámaras estáticas, como los descritos en (Zhang, 2000), (Heikkila, 2000), o (Jones et al., 2002). Sin embargo, estos sistemas de calibración de cámaras estáticas no son aplicables a las cámaras PTZ, ya que sería necesaria la calibración para cualquier posición de la cámara (en términos de posición horizontal, vertical, y zoom). Sin embargo podemos encontrar algunos trabajos recientes que tratan específicamente con cámaras PTZ, como los descritos en (Possegger et al., 2012), (Galego et al., 2012), y (Puwein et al., 2012). En este sentido, la calibración de las cámaras PTZ también difiere de las técnicas convencionales aplicadas a cámaras estáticas, pero podemos encontrar métodos específicos para tratar con ellas.

#### 2.4.2.4 Seguimiento de objetos

Los algoritmos de segmentación sirven únicamente para segmentar objetos o patrones de interés en una imagen o secuencia de imágenes capturadas por una cámara. Sobre esta salida necesitaríamos extraer las características de los objetos como su tamaño, color, flujo óptico, forma, textura u otras características y añadir la componente temporal. Esto nos permitirá asociarles a los objetos una identidad, y determinar su evolución en el tiempo, como puede ser su trayectoria. Hay numerosos métodos y algoritmos que nos permiten seguir a los objetos a lo largo del tiempo, y básicamente suelen consistir en encontrar las correspondencias entre dos elementos detectados en imágenes sucesivas (Yilmaz et al., 2006). Se podrían destacar tres grandes bloques de métodos:

**Seguimiento basado en puntos:** Los objetos detectados en imágenes consecutivas se representan con una serie de puntos, y la asociación de los puntos entre imágenes se basa en el estado previo de los objetos, como su posición y su movimiento. Hay dos enfoques principales, el determinista y el probabilista. En los métodos deterministas se suele calcular la puntuación de asociación utilizando uno o varios de los siguientes factores: proximidad, aceleración, velocidad máxima, rigidez del objeto y otras restricciones de movimiento. Los métodos probabilistas suelen utilizar sin embargo una representación del estado compuesto de posición, velocidad y aceleración, que dependiendo de si hay uno o varios objetos a seguir, se suelen aplicar algoritmos de filtrado como el filtro de Kalman (Broida and Chellappa, 1986), filtro de partículas (Isard and Blake, 1998), el filtro JPDA (Rasmussen and Hager, 2001), o el MHT (Cox and Hingorani, 1996).

**Seguimiento basado en la forma y apariencia:** También conocida como Kernel Tracking en inglés. En este caso los objetos pueden describirse por una forma determinada, como un rectángulo o una elipse con un histograma determinado. La asociación se realizaría comparando estas características teniendo en cuenta el movimiento que puede tener el objeto, como desplazamientos, rotaciones, o cambios de escala (Comaniciu et al., 2003; Pérez et al., 2002).

**Seguimiento basado en el contorno:** Dado el modelo de un contorno, las siluetas se asocian o bien buscando correspondencias por la forma del contorno (Shotton et al., 2005), o estimando la evolución del contorno a lo largo del tiempo (Yilmaz et al., 2004).

Además de estos algoritmos, se puede plantear el uso de múltiples cámaras para realizar el seguimiento, que puede ayudar a obtener más precisión, ya que se podría contar con más información, como puede ser la profundidad, y además se pueden llegar a resolver problemas como la oclusión (Mittal and Davis, 2003).

Estos algoritmos nos permiten por tanto procesar las imágenes de las cámaras para detectar y seguir elementos de interés dentro del entorno. Esta información puede ser utilizada en tiempo real por diversas capas de más alto nivel para muchas tareas, como puede ser la gestión de alarmas, la coordinación de cámaras para cubrir el máximo número de elementos, maximizar el número de cámaras siguiendo un elemento, y en general diversas tareas de gestión.

### 2.4.3 Arquitecturas

Podemos encontrar diferentes enfoques a nivel de arquitectura que se han ido adoptando a lo largo del tiempo. Entre ellos podemos encontrar las arquitecturas centralizadas, distribuidas, híbridas y en múltiples capas. La arquitectura a utilizar se debería seleccionar en función del dominio del problema, y atendiendo a determinados parámetros como la escalabilidad o la

facilidad de mantenimiento y despliegue. A continuación se comentarán diferentes arquitecturas comúnmente aplicadas en el control de múltiples cámaras.

**Arquitecturas Distribuidas**

Las arquitecturas distribuidas consisten en un conjunto de cámaras que actúan como nodos independientes, pero que se transmiten información con nodos vecinos con el fin de llevar a cabo una tarea común de vigilancia. En este tipo de sistemas, cada nodo acaba contando con una visión global del entorno que le permite tomar decisiones de manera local. Algunas de estas arquitecturas las podemos encontrar en (García et al., 2005; Hodge and Kamel, 2003; Li and Bhanu, 2008; Qureshi and Terzopoulos, 2008; Song et al., 2008). La ventaja de estas arquitecturas distribuidas es que son altamente escalables, permitiendo despliegues con un gran número de cámaras cubriendo mayores áreas de vigilancia. También son más robustas frente a fallos, ya que el fallo de un simple nodo podría empeorar el rendimiento del sistema, pero no lo dejaría inutilizable. Por el contrario es más complejo de desarrollar, gestionar, y depurar.

**Arquitecturas Centralizadas**

Una arquitectura centralizada consiste en un nodo centralizado (un sistema informático) que controla un conjunto de nodos (cámaras) subordinados para que realicen las tareas de vigilancia. Normalmente este nodo centralizado recibe toda la información del resto de cámaras, la procesa, fusiona la información, toma decisiones, y planifica el control sobre el resto de nodos. Evidentemente este tipo de arquitecturas son muy poco escalables y pueden presentar problemas en despliegues con un número significativo de cámaras. También sería difícil hacer un sistema robusto con esta arquitectura, pues el fallo del nodo central podría hacer que el sistema entero se parase. Por el contrario tienen la ventaja de que son más fáciles de implementar, ya que toda la información y la lógica de control se encuentra centralizada en un único nodo, simplificando así el proceso de desarrollo e integración.

**Arquitecturas híbridas**

Adicionalmente podemos encontrar arquitecturas híbridas, que resultan de una combinación entre una arquitectura centralizada y otra híbrida. Por ejemplo, en esta arquitectura podríamos contar con una serie de nodos que procesen la información de más bajo nivel, como la detección, seguimiento y clasificación, descargando al nodo central del procesamiento. De este modo el nodo central sólo necesitaría procesar y fusionar la información para obtener más información o bien tomar decisiones de alto nivel, como el trabajo descrito en (Gualdi et al., 2008). En este caso se tendrían las ventajas de las arquitecturas centralizadas pero descargando de forma notable al nodo central. Pero de igual modo sigue siendo poco escalable y poco robusto frente fallos.

**Arquitecturas en múltiples capas**

Las arquitecturas basadas en múltiples capas son una variante de las arquitecturas híbridas, donde se suelen definir varios niveles de interacción o jerarquías entre nodos. Por ejemplo, en el trabajo presentado en (Ukita and Matsuyama, 2005) se propone una arquitectura de tres capas donde los nodos superiores son encargados de controlar los nodos de más bajo nivel a través de una capa intermedia que los agrupa. En este caso se suelen organizar los nodos por funcionalidades, donde cada uno tiene un rol determinado y necesita interactuar con otros nodos que le proporcionen otras funcionalidades.

### 2.4.4 Gestión basada en sistemas multi-agente

Los sistemas multi-agente (MAS) consisten en agentes autónomos que son capaces de percibir información del entorno y además actuar sobre él. Estos agentes son capaces de interactuar entre ellos con el fin de solucionar problemas que posiblemente no se podrían resolver con agentes independientes. Un agente se suele asociar a diversas entidades como componentes software, humanos, robots, sensores, etc. En entornos de video vigilancia, los agentes normalmente se refieren a cámaras estáticas (Hodge and Kamel, 2003; Monari et al., 2008), cámaras PTZ (Kim et al., 2006), Smart Cameras (Bramberger et al., 2005), y otros sensores (Castanedo et al., 2006). Los sistemas multi-agente proporcionan por tanto una arquitectura base que permite fácilmente desarrollar un sistema donde controlar y coordinar cámaras autónomas distribuidas. Esto es en parte gracias al modelado BDI (creencias, deseos e intenciones), que permiten incorporar objetivos de alto nivel en los agentes. Las creencias de un agente representan el conocimiento sobre las capacidades del propio agente, las capacidades de los agentes vecinos, e información sobre el entorno adquirida a través de sus sensores o proporcionada por otros agentes. Los deseos representan el objetivo final del agente, mientras que las intenciones son los pasos básicos que el agente puede realizar para llegar a cumplir sus deseos.

(Hodge and Kamel, 2003) presentó un mecanismo de coordinación basado en agentes cuyo objetivo consistía en cubrir la visibilidad de objetos estáticos con cámaras en movimiento. En esta propuesta, las cámaras estaban montadas sobre agentes móviles cuyo objetivo consistía en monitorizar estos objetos. Comunicando los deseos e intenciones de los agentes vecinos, cada agente infiere un modelo de creencias y conoce las preferencias de los agentes cercanos. Para decidir cualquier acción del agente, el algoritmo de coordinación evalúa cada posible acción y selecciona la que mejor se adapte al objetivo mediante funciones de fitness. Por el contrario, en (García et al., 2005) se planteó un sistema basado en agentes para lidiar con el problema de la coordinación entre cámaras estáticas para tareas de vigilancia como el seguimiento de objetivos entre múltiples cámaras, y la resolución de conflictos. En este caso, las cámaras eran capaces de alertar y negociar con las cámaras vecinas dependiendo de las percepciones de cada una de ellas. (Bramberger et al., 2005) presentó un sistema basado en agentes que se

encargaba de monitorizar el tráfico en largas autopistas o túneles. La particularidad de este proyecto es que los nodos de la arquitectura distribuida estaban compuestos principalmente por Smart Cameras, lo que suponía que tenían una serie de recursos limitados. El sistema por tanto sólo ejecutaba los algoritmos de seguimiento en aquellos nodos que pudiesen contribuir al seguimiento de un objetivo. En este aspecto, los agentes de seguimiento podían irse moviendo entre los distintos nodos desplegados por el sistema, realizando de esta forma un seguimiento continuo del objetivo.

(Kim et al., 2006) propone un sistema compuesto por agentes cámara (CA) y un módulo de soporte (SM). Las cámaras PTZ están dispuestas en un pasillo y realizan un seguimiento mono-objetivo que permite extraer las coordenadas 3D del objetivo. Para mejorar la precisión de la localización, los CA transmiten la información detectada al SM, que notifica a otras CA para que se comience un seguimiento estéreo. En este punto los CA deben orientar la cámara hacia el nuevo objetivo mediante posiciones PTZ preestablecidas. Ahora, con la información de ambas cámaras, el SM es capaz de proporcionar una información de posición 3D más precisa. Una arquitectura MAS similar se presentó en (Castanedo et al., 2006) para la coordinación de agentes cooperativos (CSA) usando un protocolo de coalición. Cada CSA se inicia con unas capacidades y restricciones dependiendo de la naturaleza del sensor que está controlando. Diferentes CSAs pueden cooperar entre sí para mejorar el rendimiento o tener mayor precisión en su tarea, y también para usar capacidades de otros sensores, aumentando sus capacidades y realizando tareas que no podría hacer individualmente. El protocolo de coalición consiste en sencillos mensajes como iniciar-coalición, aceptar-coalición, rechazar-coalición, informar-coalición, y dejar-coalición, que permiten a los agentes dialogar para formar coaliciones temporales.

Siguiendo con estos trabajos, (Monari et al., 2008) presentó un trabajo especializado en cubrir grandes áreas mediante cámaras distribuidas. Un sistema que implique un gran despliegue de cámaras necesita ser distribuido y escalable, debido a la gran cantidad de datos que se pueden generar. En este sentido se diseñó una arquitectura en tres capas: La capa de más bajo nivel se encarga de procesar el vídeo de las cámaras, realizando la detección, segmentación y extracción de características de los elementos. La capa intermedia, es una capa lógica que se crea y modifica bajo demanda, que básicamente consiste en agrupaciones temporales de agentes de cámaras para realizar una tarea en concreto. Estas agrupaciones se crean automáticamente dependiendo de la posición del objetivo a seguir, y también se modifican automáticamente añadiendo o quitando agentes innecesarios. Esta agrupación realiza un procesado de más alto nivel, como la fusión de datos de las cámaras que pertenecen a la agrupación. La capa de más alto nivel sirve principalmente como interacción con el usuario, permitiendo encolar tareas de vigilancia (como seguir a un objetivo determinado), que finalmente crearán nuevas agrupaciones lógicas de agentes que se encargarán de la tarea.

Más recientemente, en línea con el enfoque que seguirá esta tesis, podemos encontrar una propuesta inicial (Luis Bustamante et al., 2014a) que especifica una arquitectura multi-agente

para el control de cámaras activas y el uso de otro tipo de sensores. Estos sensores se encuentran distribuidos y cada uno está asociado a un agente específico que procesa la información generada y expone las capacidades del sensor. La información proporcionada por estos agentes, como por ejemplo la posición de los objetos detectados, es fusionada en un primer nivel dependiendo del solapamiento de los sensores en el entorno, de forma muy similar al trabajo de (Monari et al., 2008) y el uso de agrupaciones lógicas. En este caso, la fusión de información permite a los diferentes agentes detectar posibles condiciones que desencadenen diferentes tareas. Por ejemplo, un evento que indica una fusión de información de dos fuentes distintas, podría desencadenar la actuación de los agentes de control para comenzar a monitorizar diferentes zonas, y así cubrir otras áreas de vigilancia. O por el contrario, los agentes de control podrían querer maximizar la información fusionada, obteniendo así una mejor precisión e información sobre un objetivo concreto. El comportamiento de estos agentes de control dependería de los objetivos establecidos por el operador. En esta arquitectura también se contemplan otros agentes de más alto nivel, como agentes para la interfaz del usuario, agentes que registren eventos, o incluso agentes que graben el histórico de los objetos seguidos con el vídeo de múltiples vistas, ubicaciones, etc.

Como conclusión, los sistemas multi-agente proporcionan una buena plataforma para modelar el problema de la coordinación y control de múltiples cámaras en un entorno de vigilancia. La existencia de diferentes agentes con las habilidades necesarias para coordinarse son útiles para atajar problemas que no sería posible solucionar con un único agente. Sumado a esto, se pueden encontrar diversas plataformas software sobre las que especificar y modelar el comportamiento de los agentes, la comunicación, el despliegue, etc., lo que simplifica enormemente su desarrollo en un entorno real.

### 2.4.5 Gestión basada en la teoría de control

La teoría de Control es un campo interdisciplinario de la ingeniería y las matemáticas que trata con el comportamiento de sistemas dinámicos con entradas, y cómo este comportamiento se modifica con una retroalimentación. El objetivo habitual en la teoría de control es manejar un sistema de tal forma que su salida siga una señal de control específica, habitualmente llamada referencia, que puede ser una señal constante o variable. Para conseguir este comportamiento, el controlador monitoriza constantemente la salida del sistema y la compara con la referencia. La diferencia entre la salida actual y la salida deseada se incluye como entrada al sistema en forma de retroalimentación. De esta forma el sistema puede ajustar su comportamiento para acercarse cada vez más a la salida deseada, minimizando el error medido. Hay controladores que pueden ser únicamente proporcionales (P), proporcional integral (PI), y proporcional, integral y derivativo (PID). El controlador P es el único que no tiene en cuenta la variación del error en el tiempo, y suelen ser controladores que acumulan mayor error en estado estacionario.

Este tipo de sistemas se ha empleado en alguna investigación con el fin del controlar cámaras PTZ que realicen el seguimiento de regiones de interés. Por ejemplo, en (Al Haj et al., 2010) se utiliza un controlador PID para controlar el movimiento de la cámara con el objetivo de centrar en la cámara los objetos de interés. El filtro PID toma en este caso como entrada la salida de un filtro EKF (Filtro de Kalman Extendido) que filtra las posiciones del objeto detectado. (Singh et al., 2008) planteó un control de cámaras activo y cooperativo para seguir rostros usando un Modelo de Control Predictivo (MPC) como mecanismo de retroalimentación. El controlador MPC no sólo tenía en cuenta el error actual y pasado como un controlador PID, si no que además añade otra variable como el error futuro que se estima que tendrá, basándose en la salida de un filtro de Kalman. En este trabajo se compara el controlador MPC contra un PID tradicional, sobre el cual se observa que obtienen una ligera ventaja. Además se define un modo de interacción entre cámaras basado en la cooperación y competición, que permite intercambiar roles y sub-objetivos, mejorando así el desempeño del sistema. (Gans et al., 2009) También usa un controlador PID para mantener centrados en el campo de visión de la cámara múltiples objetivos simultáneamente. En este caso no se trata de una arquitectura multi-cámara, si no más bien un controlador de bajo nivel que permite ajustar la cámara para centrar los diferentes objetivos.

En general, los enfoques basados en la teoría de control son útiles principalmente para reducir el error cuando se controlan sensores, como puede ser en este caso manejar las cámaras PTZ para seguir caras u objetos. Sin embargo tienen grandes limitaciones en el modelado de entornos más complejos donde varios sensores puedan colaborar e interactuar entre sí. Probablemente el uso de estos sistemas esté limitado a estas tareas de más bajo nivel, por lo que aún sería necesario sistemas de más alto nivel si queremos coordinar cámaras, tal y como se describe en (Singh et al., 2008).

### 2.4.6 Gestión basada en la teoría de la decisión

La teoría de la decisión proporciona un marco de trabajo para seleccionar una acción cuando las consecuencias como resultado de diferentes elecciones no son perfectamente conocidas. La teoría de la decisión se fundamenta en la teoría de la utilidad y el uso inductivo de la teoría de la probabilidad. Básicamente consiste en un conjunto de técnicas matemáticas para tomar decisiones óptimas en entornos estocásticos o con incertidumbre. Proporcionan métodos formales como el Proceso de Decisión de Markov (MDP), y el Proceso de Decisión de Markov Parcialmente Observable (POMDP), que pueden ser utilizados para controlar y coordinar múltiples elementos (como cámaras y sensores) en entornos de vigilancia. Estos entornos suelen estar cargados de múltiples fuentes de incertidumbre, como la dinámica y posición de los objetivos, cámaras con observaciones ruidosas, sensores de localización imprecisos, etc. Un sistema basado en la teoría de la decisión, podría tener en cuenta estas incertidumbres a través

de modelos probabilísticos.

En este ámbito podemos encontrar diferentes trabajos que hacen uso de la teoría de la decisión, como el propuesto en (Spaan and Lima, 2009). En este trabajo se busca seleccionar las cámaras estáticas óptimas de un entorno de vigilancia, basándose en los objetivos definidos por el usuario, como por ejemplo, el seguimiento de un objetivo en particular. (Daniyal and Cavallaro, 2011; Daniyal et al., 2010) por su parte proponen un proceso POMDP que les permita seleccionar automáticamente la cámara que mejor represente el estado del entorno, en este caso para la visualización de eventos deportivos (podemos tener múltiples cámaras, pero el espectador sólo puede ver una de forma simultánea). Este tipo de enfoque se podría extrapolar fácilmente a los entornos de vigilancia donde los operadores no pueden muchas veces monitorizar simultáneamente todas las cámaras disponibles. Para ello emplean un modelo de distribución gaussiana multivariante con el que medir la calidad de la vista de cada cámara, y seleccionar aquella con mayor puntuación y que implique el menor número de cambios de vista en el futuro.

Más recientemente (Natarajan et al., 2012a) y (Natarajan et al., 2012b), presentaron propuestas basadas en MDP y POMDP para controlar y coordinar múltiples cámaras PTZ. Para ello se modeló la tarea de vigilancia como un sistema de optimización estocástico donde los movimientos óptimos de las cámaras PTZ se determinan intentando maximizar el número de objetos visualizados con un mínimo de resolución. En (Natarajan et al., 2012a) se utiliza una cámara estática que es la que determina la posición y velocidad de los objetos, mientras que son las cámaras PTZ las que visualizan con mayor resolución los elementos detectados. Posteriormente se eliminó la necesidad del uso de la cámara estática (Natarajan et al., 2012b), pero a costa de relajar el problema limitando el número de posiciones de las cámaras PTZ, de tal forma que las tres posiciones predefinidas podían ser calibrarlas con algoritmos convencionales.

En resumen, la teoría de la decisión puede ser ventajosa cuando es necesario seleccionar una decisión óptima en presencia de incertidumbre en el entorno de vigilancia. Esta teoría encaja principalmente en aplicaciones donde es necesario modelar las interacciones entre los sensores y el entorno. Se asume que no hay otros agentes externos tomando las decisiones. El problema con la teoría de la decisión es su escalabilidad en término de número de sensores y objetivos. Esto es, cuando el número de sensores y objetivos incrementa, el espacio de estados incrementa exponencialmente y computar las decisiones óptimas puede llegar a ser inabordable.

### 2.4.7 Gestión basada en la teoría de juegos

La teoría de juegos es una rama de la teoría de la decisión que trata con decisiones interdependientes mientras que se optimiza el objetivo deseado. Un juego en la teoría de juegos es un objeto matemático que consta de un conjunto de jugadores, movimientos para los jugadores, y una especificación de recompensas para cada combinación de movimientos. La solución del

juego consiste en identificar la secuencia de movimientos que los jugadores deben seguir. Una secuencia de movimientos suele llamarse estrategia, por lo que una estrategia óptima es un conjunto de movimientos que proporcionan el mejor resultado. La teoría de juegos de centra principalmente en las interacciones entre jugadores, mientras que la teoría de la decisión modela la interacción de los jugadores con el entorno. La teoría de juegos se usa en muchos problemas de vigilancia, como el seguimiento cooperativo y competitivo, donde las cámaras se modelan como jugadores y el objetivo del juego consiste en seguir objetos. La estrategia óptima del juego consiste en seleccionar las acciones de las cámaras de tal forma que el sistema de vigilancia mejore su rendimiento (visualice más objetos, con más resolución, etc).

Recientemente (Song et al., 2011) describió las líneas generales a seguir para adaptar la teoría de juegos en una sistema multi-cámara distribuido. Para ello toman en consideración varios aspectos útiles para el operador, como la optimización del seguimiento de objetivos, la calidad/resolución de las imágenes tomadas, el área de vigilancia cubierto, etc.

(Song et al., 2008) presentó un algoritmo de control de cámaras descentralizado que permitía un seguimiento de objetivos de forma precisa y eficiente usando un enfoque de teoría de juegos. El objetivo de su sistema es seguir todos los posibles objetivos en una escena y selectivamente enfocar determinados objetivos a diferentes resoluciones. Los autores dejaron patente la necesidad de las cámaras activas, y de una estrategia de control cooperativa. La estrategia de control cooperativa se adoptó porque cada cámara tiene sus propios parámetros (orientación, campo de visión, etc), lo que impone ciertas restricciones a las otras cámaras cuando se tiene que satisfacer un objetivo. Por ejemplo, cuando una cámara enfoca un objetivo, ésta pide a las cámaras vecinas que obtengan una vista panorámica de la escena para evitar el riesgo de perder otros objetivos. En esta arquitectura distribuida, cada nodo tiene un módulo de decisión local y un sistema de retroalimentación para controlar la cámara. Cada cámara toma sus decisiones basándose en el análisis de la información que ella misma percibe a través de sus sensores y las negociaciones realizadas con otras cámaras.

(Soto et al., 2009) diseñó una estrategia distribuida para una red de cámaras PTZ donde se toman decisiones locales en cada nodo, pero donde todos los nodos se encuentran alineados para conseguir el mismo objetivo global: observar múltiples objetivos a diferentes resoluciones. Para ello se basan en la teoría de juegos, y más concretamente en mecanismos de aprendizaje y negociación multi-jugador. Las cámaras se modelan como un equipo donde cada cámara es un jugador que anota un tanto cada vez que consigue obtener una imagen de un objetivo a una determinada resolución. Estas cámaras tienen que negociar basándose en funciones de utilidad locales de tal forma que la selección de un conjunto de objetivos cumpla el equilibrio de Nash.

Li y Bhanu también propusieron una serie de trabajos basados en la teoría de juegos. Por ejemplo, en (Li and Bhanu, 2011a) realizaron una comparación entre diferentes mecanismos de selección y control de cámaras necesarios para realizar el seguimiento de objetivos. Los autores presentaron experimentos teóricos y análisis experimentales de enfoques geométricos,

estadísticos, y basados en la teoría de juegos para el control de cámaras en entornos centralizados y distribuidos. En (Li and Bhanu, 2011b) presentan un trabajo que trata con diferentes funciones de utilidad para el control de las cámaras: utilidad global para medir el grado de satisfacción del sistema en términos de seguimiento, utilidad de la cámara para determinar lo bien que la cámara está siguiendo a la persona asignada, y la utilidad de la persona para determinar cómo de recomendable es una persona para su seguimiento. Con el equilibrio del juego, aplicando estas funciones de utilidad, consiguen solucionar el problema de asignación de objetivos entre las diferentes cámaras (jugadores).

En general, la teoría de juegos es útil para tomar decisiones óptimas cuando dos o más entidades basan sus decisiones en la interacción/negociación con otras entidades. La mayoría de los problemas de vigilancia se pueden modelar como un juego, donde un jugador está jugando contra el entorno de vigilancia. En los sistemas de vigilancia, la teoría de juegos se suele aplicar de forma distribuida, de tal forma que se mejora la escalabilidad del sistema.

# 3

# Arquitectura Multi-Agente Para La Gestión de Sensores

## 3.1 Introducción

Hoy día los entornos que nos rodean están evolucionando hacia sistemas complejos que trabajan con múltiples sensores heterogéneos espacialmente distribuidos por el entorno. Los avances en las tecnologías, especialmente las de comunicación, permiten un despliegue más fácil, barato, y a mayor escala. Este tipo de sistemas se utiliza diariamente para la monitorización del medio ambiente, procesos industriales, sistemas de vigilancia. seguridad del hogar, seguridad pública, vigilancia marítima, de costas, de fronteras, Smart Cities, y un sin fin de entornos donde la seguridad o la monitorización es un elemento crítico.

Estos sistemas pueden llegar a contar con un gran número de sensores de diferente índole que muchas veces tienen que integrarse en conjunto para conseguir un objetivo de monitorización o vigilancia determinado. Podríamos contar con sensores de visión, de localización, de presencia, de parámetros ambientales, etc. La integración de todos los sensores puede llegar a suponer un gran reto, por ejemplo si contamos con sensores de diferentes fabricantes con diferentes protocolos. Un despliegue específico podría satisfacer un entorno determinado, pero no se adaptaría bien a nuevos sensores u otros entornos.

Además de sensores, en este tipo de entornos es muy habitual encontrar actuadores o sistemas que requieran de cierta gestión. Por ejemplo, a nivel físico podríamos pensar en actuadores que interactúan con el entorno como alarmas, motores, relés, o cámaras PTZ. Por otro lado, a nivel lógico, podríamos considerar la necesidad de gestionar o ajustar otro tipo de parámetros que no están directamente relacionados con la modificación del entorno. Por ejemplo, podríamos contar con sistemas desplegados en el entorno que necesiten actualizar determinados parámetros en función de las condiciones cambiantes del entorno. En diferentes proyectos con empresas hemos podido constatar estas necesidades por ejemplo con los sistemas

de radar, que deben ser configurados en función de las condiciones meteorológicas del momento, ya que estos no se comportan igual con lluvia, niebla, e incluso olas.

De este modo contamos con entornos que contienen multitud de sensores heterogéneos, que pueden estar distribuidos espacialmente por el entorno, y que además pueden requerir de cierta gestión o de control. Debido a la gran cantidad de sensores y actuadores que suele haber presentes en este tipo de entornos, surge la necesidad de poder desplegar algún tipo de sistema que permita la integración y la gestión autónoma de los diferentes elementos.

En este sentido podemos contemplar el uso de los Agentes. Por definición (Russell et al., 1995), un agente puede ser visto como cualquier elemento que se encuentra monitorizando su entorno a través de sus sensores, y actúa sobre él a través de sus actuadores. Podemos ver una imagen muy representativa de esta descripción en la figura 3.1. Esta definición de agente casaría muy bien con la problemática de gestión multi-sensor que estamos describiendo. Ya que necesitamos un conjunto de entidades que puedan estar monitorizando el entorno y realizando algún tipo de gestión autónoma.

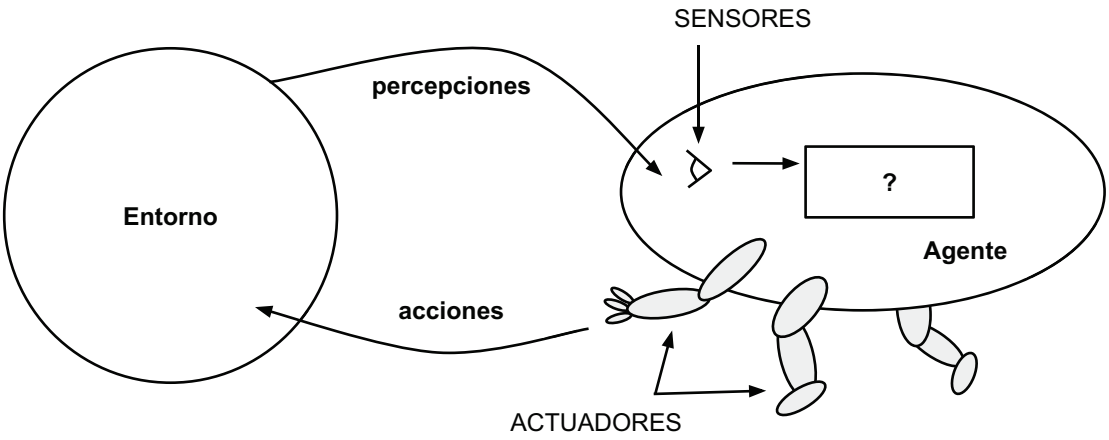


Figura 3.1: Descripción de un agente. Este puede ser visto como una entidad que monitoriza el entorno a través de sus sensores, y actúa sobre él a través de sus actuadores.

Si bien esta definición de agente es bastante acertada para el problema a resolver, los agentes como entidades independientes podrían quedarse limitados para según qué tipo de tareas. En este tipo de entornos podríamos encontrarnos con problemas si esta gestión necesita de algún tipo de coordinación. Recordemos que son entornos distribuidos que pueden cubrir grandes áreas de monitorización.

Sería posible pensar en arquitecturas centralizadas que recojan toda la información del entorno, la procesen, y decidan la gestión a aplicar en cada momento. Sin embargo, estas arquitecturas fallarían en escalabilidad, tolerancia a fallos, e integración. Sería más apropiado pensar en arquitecturas distribuidas, que tienen una relación más natural con este tipo de

entornos.

Por ello, además de pensar en el término de agente, podemos pensar en un concepto más global como el término de sistemas multi-agente. Un sistema multi-agente se puede ver como un sistema informático compuesto de múltiples agentes inteligentes que interactúan en un entorno. Ya no sólo contamos con agentes independientes, si no que estos agentes desplegados por el entorno podrían tener capacidades de comunicación, y por tanto, de coordinación. Podemos ver una representación de estas arquitecturas en la figura 3.2. En esta aparecen múltiples agentes monitorizando y gestionando el entorno, pero además interaccionando entre ellos.

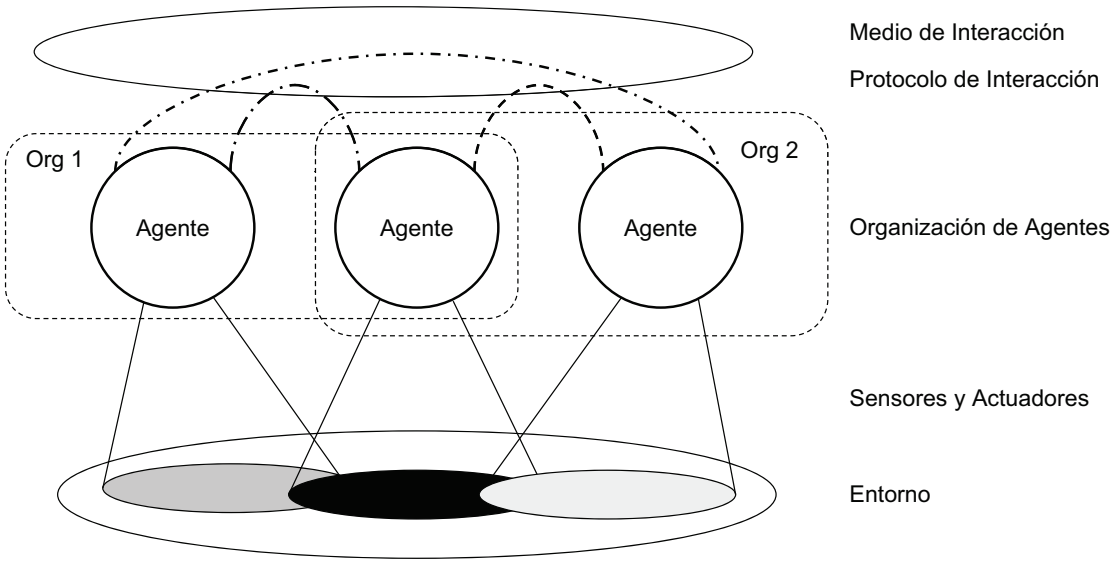


Figura 3.2: Descripción de un sistema multi-agente. En este tipo de entornos podemos encontrar múltiples agentes distribuidos por el entorno con la capacidad para comunicarse y coordinarse para solucionar diferentes tareas.

Por tanto, en esta sección se describe el diseño del sistema multi-agente genérico que permita la integración de múltiples sensores heterogéneos para su gestión y coordinación de forma autónoma. Este diseño se centrará principalmente en la definición de los diferentes agentes identificados, así como la información de alto nivel que se considera que necesitarán transmitirse.

## 3.2 Metodología

Una vez vista la problemática a resolver, y teniendo en cuenta la variedad de sensores y actuadores que nos podemos encontrar en estos entornos, es hora de diseñar la arquitectura que permita su integración y gestión. Para ello nos basaremos en una arquitectura multi-agente, debido principalmente a las ventajas descritas en la sección 2.4.4. Esto nos permitirá diseñar el problema como un sistema distribuido que cuenta con diferentes entidades con la capacidad de comunicarse y colaborar.

Para ello será necesario definir las diferentes entidades, el rol que desempeña cada una, la información que se transmitirán, etc. Es obvio que cada entorno contará con multitud de particularidades relacionadas con los sensores, su distribución, la información que generan, qué gestión o coordinación necesitan, etc. Por ello, en esta sección se propondrá un diseño de alto nivel que represente el dominio con el que estamos trabajando. Los detalles de implementación o aplicación a cada entorno particular, lamentablemente deberán diseñarse y adecuarse para satisfacer los requisitos que puedan aparecer. Por este motivo, tras el diseño general propuesto en esta sección, en las secciones 4 y 5, veremos su adecuación a dos entornos particulares que cuentan con sensores, información y objetivos de gestión diferentes.

El diseño del sistema multi-agente a este nivel, por tanto, consistirá en la identificación de los diferentes agentes, la definición de sus objetivos o roles, la relación con el resto de agentes, y la información que se espera que se transmitan. Este nivel de diseño se correspondería con el primer paso de la metodología de diseño de sistemas multi-agente descrita en (DeLoach, 1999), que pude verse en la figura 3.3. Como podemos observar, el primer paso consistirá en el diseño a nivel de dominio. Esta base de diseño servirá para el desarrollo de la siguientes etapas que deberían ser particularizadas y optimizadas para cada problema en concreto.

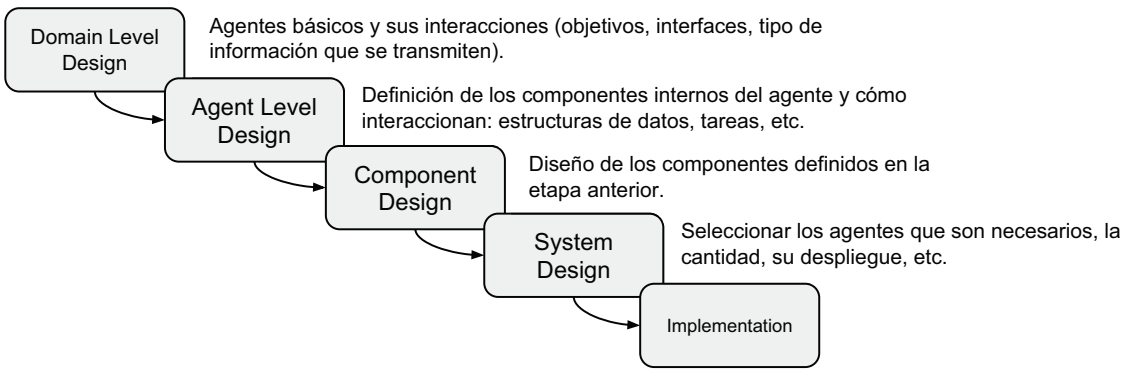


Figura 3.3: Para el diseño del sistema multi-agente general nos centraremos en el primer paso de la metodología, esto es, el diseño a nivel de dominio, donde básicamente se describen los agentes identificados, sus objetivos, y la información que se transmiten.

## 3.3 Diseño del sistema Multi-Agente

Podemos considerar el sistema multi-agente como un elemento de gestión intermedia entre el operador y el entorno. Por un lado el sistema multi-agente interactúa con el entorno a través de los sensores. Esto les permite percibir la información que se genera. Por otro lado podemos contar con el operador que monitorizaría y controla el proceso de gestión. El sistema multi-agente debería ser capaz de gestionar los sensores u actuadores en función de la información percibida, y las tareas u objetivos establecidos por el operador. Se puede ver una representación de esta interacción en la figura 3.4.

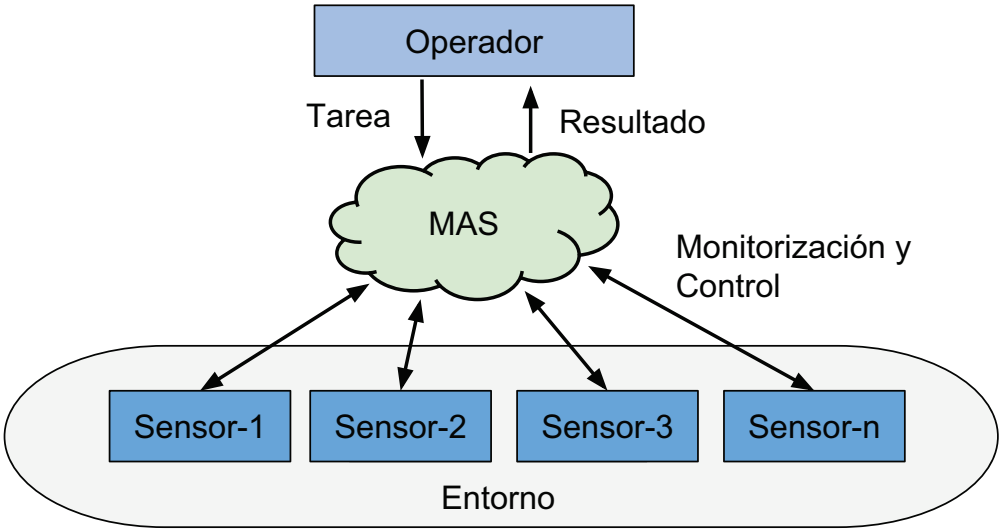


Figura 3.4: Visión general del sistema multi-agente y su interacción con los sensores y el usuario.

En este sección pasaremos a describir los diferentes agentes implicados dentro de esta arquitectura multi-agente que permitirá la gestión de múltiples sensores u actuadores.

### 3.3.1 Agente Sensor/Actuador

El Agente Sensor (*SA*), de bajo nivel dentro de la arquitectura, se encarga de representar al sensor o actuador dentro de la arquitectura multi-agente. Estos dispositivos, encargados de monitorizar o realizar cambios en el entorno, por sí solos no pueden integrarse fácilmente en una arquitectura multi-agente, ya que normalmente no cuentan con las mismas interfaces y protocolos de comunicación. Por lo tanto, este agente hará de interfaz entre el dispositivo físico y el resto de la arquitectura, posibilitando la comunicación a otros agentes de la arquitectura. Esto implica que este agente deberá ser adaptado a cada dispositivo particular que se integre en el sistema, ya sea una cámara de vídeo, un sensor de presencia, un radar, una electroválvula, un cierre electrónico, etc.

Los sensores y actuadores distribuidos en un entorno pueden ser de diferente naturaleza y características, por lo que suelen necesitar una integración específica. Del mismo modo, cada agente que represente un tipo concreto de sensor deberá ser diseñado específicamente para que pueda interactuar con él. Por ejemplo, si tenemos un sensor de vídeo y un sensor de presencia, necesitaremos especificar dos Agentes Sensor diferentes. El primero podría ser un agente de vídeo que capture las imágenes del dispositivo, y proporcione algún tipo de información sobre la escena visualizada (objetos, personas, actividades, seguimiento, etc). El segundo sería un agente especializado en el sensor de presencia y podría comunicar las detecciones de actividad de una zona determinada. Esta información capturada y procesada ya puede ser transmitida a otros agentes empleando un lenguaje común entre ellos.

En algunos casos podemos encontrarnos con sensores que también cuentan con actuadores, como por ejemplo una cámara PTZ. En este caso podemos controlar parámetros como la posición horizontal, vertical, o el nivel de zoom. Aquí el agente también deberá tener la capacidad de ajustar estos parámetros y ofrecer esta habilidad a otros agentes que puedan necesitar interactuar con el entorno.

Dentro del modelo de fusión JDL comentado en el estado del arte, este agente se encontraría entre los niveles 0 y 1. A nivel 0 porque este agente trataría con la información generada directamente por el sensor, realizando el procesamiento de la información dependiendo del sensor con el que se encuentre trabajando. Y a nivel 1 porque este agente debería llegar a ser capaz de generar información sobre las entidades identificadas en el entorno. Por lo que esta información podría ser utilizada por el resto de agentes trabajando en otros niveles del modelo para completar el proceso de gestión.

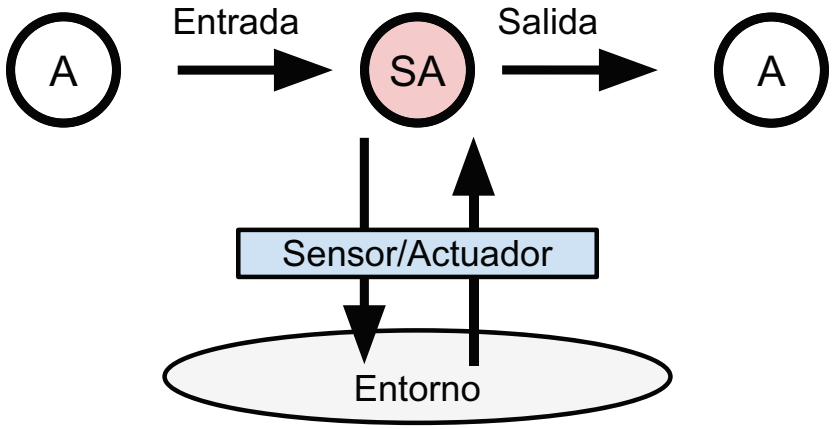


Figura 3.5: Ejemplo del agente sensor (*SA*) que interactúa con el entorno a través de su sensor asociado. Existirán otros agentes en la arquitectura que recibirán la información percibida del entorno, y otros que podrán realizar determinados ajustes.

En la figura 3.5 podemos encontrar una representación del agente sensor (*SA*). El agente

*SA* está en contacto directo con el sensor/actuador para monitorizar o realizar cambios sobre el entorno. La información recopilada por este agente puede ser transmitida en tiempo real hacia otros agentes de la arquitectura que puedan necesitar esta información. Por otro lado este agente también acepta entradas que normalmente implican el ajuste/control del sensor o actuador.

Siguiendo con el ejemplo de una cámara PTZ, por un lado tendríamos un *SA* encargado de recibir las imágenes de la cámara, analizarlas, y transmitir la información de posición de los blancos detectados en la escena. Por otro lado contaríamos con un Agente de Control (*CA*) que está recibiendo esta información en tiempo real. El agente *CA* podría contar con un algoritmo que determine el objetivo de más interés (más grande, más cercano, con una determinada forma, color, etc.) y ajuste a través del *SA* los parámetros de la cámara para centrar o realizar zoom sobre un determinado objetivo.

| **Entrada** | |
|---|---|
| **Datos del Sensor** | El sensor proporciona la información que permite monitorizar el entorno. Esta información puede ser en forma de imágenes, detecciones de presencia, ubicaciones, información de identidad, etc. El agente tendrá que gestionar esta información, y en algunos casos realizar determinados procesamientos a nivel 0 del modelo JDL. |
| **Información de Control** | Otros agentes pueden querer interactuar con el entorno, y para ello pueden solicitar al agente determinados cambios en el sensor/actuador. Por ello este agente deberá responder a las peticiones de control recibidas. |
| **Salida** | |
| **Información del entorno** | El Agente sensor deberá ser capaz de procesar la información generada por el sensor y proporcionarla a otros agentes que puedan necesitarla. La información podrá ser más rica que la proporcionada directamente por el sensor, por ejemplo, podría contener información de las trazas de una cámara de vídeo, en lugar de las imágenes en sí. Dependiendo del entorno, esta información será de nivel 0 o nivel 1 del modelo JDL. |
| **Información del sensor** | Para que otros agentes puedan interactuar con el sensor u actuador, es necesario que el resto de agentes conozcan su tipo y propiedades, por lo que el agente deberá ser capaz de ser descubierto por otros agentes y proporcionar este tipo de información. |

Tabla 3.1: Información de entrada y salida del Agente Sensor (*SA*).

De forma general, podríamos describir la entrada y salida de información de este agente tal y como se muestra en la tabla 3.1.

### 3.3.2 Agente de Fusión

En una arquitectura de sensores distribuida es posible que nos encontremos sensores monitorizando el entorno que proporcionen información redundante de alguna de las medidas. Desde el punto de vista de la información, en estos casos podremos obtener información más rica, precisa, o complementaria sobre las diferentes entidades monitorizadas. Para ello es necesario aplicar un proceso de fusión que combine toda esta información redundante o complementaria.

Este agente se encargaría de integrar la información susceptible de ser fusionada, y que es recibida directamente desde los diferentes *SA* que se encuentran monitorizando el entorno. La información fusionada podrá ser utilizada de nuevo por otros agentes de la arquitectura, que les facilitará la evaluación de la situación del entorno, la toma de decisiones, la resolución de conflictos, y en general obtener un mejor rendimiento en el desempeño de su tarea.

Este agente estaría exclusivamente enmarcado dentro del nivel 1 del modelo JDL, ya que es el agente encargado de percibir las percepciones generadas por los *SA* que ya proporcionan información de nivel 0/1, y generar una salida refinada de nivel 1. En este agente se deberían integrar los algoritmos y procesos típicos de la fusión de datos, como pueden ser los mecanismos de alineación temporal de las medidas, la aplicación de los algoritmos de asociación y filtrado, la combinación de la estimación de los estados, etc. Estos algoritmos será necesario definirlos en la especificación del *FA* dentro del entorno concreto sobre el que se vaya a trabajar, y en base a la información generada por el resto de *SA*. La información básica de alto nivel gestionada por este agente se resume en la tabla 3.2.

| **Entrada** | |
|---|---|
| **Información Nivel 0/1 JDL** | El *FA* recibirá información ya procesada, a nivel 0 o 1 del modelo JDL desde los diferentes *SA* o *FA* desplegados. Esta información será información local de cada sensor, o inclusive información ya integrada por otro *FA*. |
| **Salida** | |
| **Información Nivel 1 JDL** | La salida del *FA* consistirá en información procesada de nivel 1, donde los objetos u entidades de salida ya se encontrarían refinados e identificados, dependiendo de la información de entrada recibida. |

Tabla 3.2: Información de entrada y salida del Agente Fusión (*FA*).

Sin embargo, a este nivel de la arquitectura no es posible aún especificar los algoritmos de fusión a integrar, ya que estos dependerán completamente del dominio, de si contamos con información de posición, el modelo de incertidumbre sobre las medidas, y en general de las características particulares de los datos.

La fusión de datos dentro del sistema multi-agente la podemos organizar de diferentes formas dependiendo del número, características y disposición de los sensores, su distribución espacial, el solapamiento de las coberturas, etc. Dependiendo del entorno concreto y sus características, por tanto será necesario optar por una arquitectura de fusión u otra. De nuevo en este caso no se limita el uso de una arquitectura u otra, ya que esta decisión dependerá del diseño particular sobre el que se esté trabajando.

En las siguientes subsecciones se describen las diferentes arquitecturas de fusión que se podrían plantear dentro del sistema multi-agente.

#### 3.3.2.1 Fusión Centralizada

En una arquitectura centralizada cada sensor del entorno (*SA*) proporciona un conjunto de medidas a un Agente Fusión (*FA*). El *SA* es capaz de generar un conjunto de medidas de forma independiente. Estas medidas procedentes de los diferentes sensores son alineadas espacial y temporalmente (dependiendo del tipo de información) y asociadas en una única entidad fusionada que combina su estado (por ejemplo, la posición de una determinada entidad). La estimación de la entidad fusionada será más refinada cuanto mayor sea el número de aportaciones de aportaciones de los sensores.

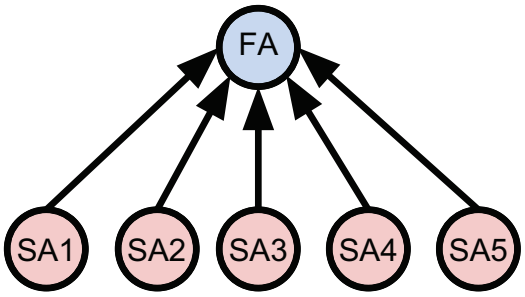


Figura 3.6: Arquitectura de fusión centralizada. Múltiples agentes sensor (*SA*) proporcionan medidas a un único agente de fusión (*FA*) que integra todas las medidas.

El *FA* se encarga de integrar la información y proporcionar un conjunto de entidades no redundantes entre los sensores. Los sensores pueden proporcionar la información de forma asíncrona o con un intervalo constante de actualizaciones. El agente de fusión, u otros agentes que dependan de él, pueden necesitar actualizaciones de la información incluso cuando no se reciban medidas de los sensores, por lo que es necesario que este agente sea capaz de

realizar estimaciones sobre las entidades fusionadas. Las estimaciones de las entidades se generan directamente sobre las medidas de los sensores, por lo que se puede considerar que son estimaciones óptimas.

En la figura 3.6 se puede observar una representación de una arquitectura de fusión centralizada, donde varios agentes sensor proporcionan medidas a un único agente de fusión. La salida de este agente fusión se considerará una salida de fusión global, ya que toda la información complementaria ha sido integrada, proporcionando un conjunto de entidades no redundantes.

#### 3.3.2.2 Fusión Jerárquica

La arquitectura de fusión jerárquica podría considerarse como una extensión de la arquitectura de fusión centralizada donde podemos encontrar diferentes niveles de fusión. Como se muestra en la figura 3.7, también contamos con un conjunto de sensores que envían sus medidas a un *FA* local. En este caso, estos agentes de fusión ($FA1$ y $FA2$) realizan una fusión local. Las estimaciones de las entidades fusionadas en este primer nivel son a su vez transmitidas a un nivel superior donde se realiza una fusión global para generar una estimación conjunta.

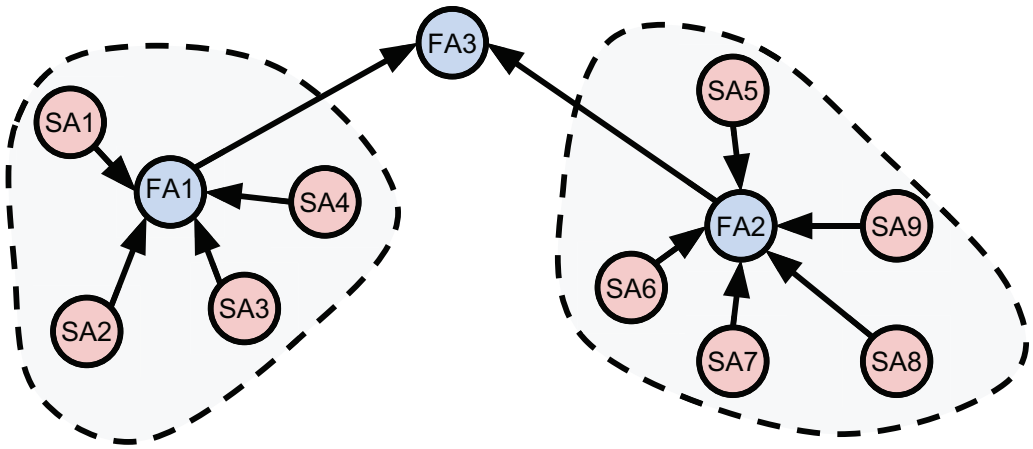


Figura 3.7: Esquema de distribución de los agentes de fusión en entornos donde las áreas de solapamiento permitan distribuir la carga de procesamiento de un nodo centralizado de fusión

Tiene sentido aplicar una arquitectura de fusión jerárquica cuando contamos con sensores de diferente naturaleza o que cubren diferentes zonas que no se solapan. En el nodo de fusión local se procesarían las medidas de un grupo determinado de sensores y sólo se transmitiría la información ya refinada y fusionada a los nodos de fusión superiores. Por este motivo, las

arquitecturas jerárquicas son útiles en entornos donde nos encontramos restricciones en el ancho de banda, como puede ser la cantidad (y calidad) de los datos generados de los nodos locales.

#### 3.3.2.3 Fusión Distribuida

Una arquitectura de fusión distribuida no asume ningún rol entre los nodos de fusión, como puede ser la fusión local o global. En este caso la transmisión de información entre los nodos se realiza entre iguales. Los datos son transmitidos según las necesidades de cada nodo, por ejemplo, requiriendo información de otros nodos para ayudar a reducir la incertidumbre local de una entidad, o bien enviando información en momentos relevantes, como la creación, actualización, o borrado de entidades.

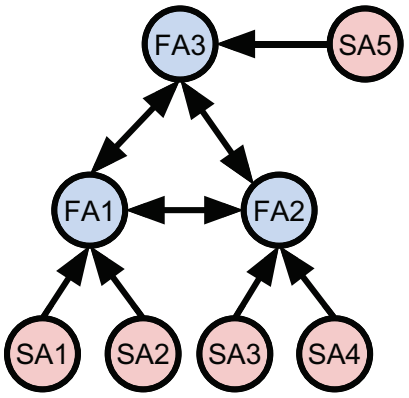


Figura 3.8: Arquitectura de fusión distribuida. Los agentes de fusión (*FA*) se transmiten información según sea necesario. No tiene por qué haber un único punto de fusión, ya que la información fusionada podrá estar distribuida entre diferentes nodos según sea necesario.

### 3.3.3 Agente Control

En el caso de que los sensores o actuadores del entorno requieran de algún tipo de intervención autónoma es necesario desplegar Agentes de Control (*CA*) que se hagan cargo de esta tarea. Por ejemplo, si nos encontramos en un entorno de videovigilancia con un conjunto de cámaras PTZ, un *CA* podría ser aquél que monitoriza y controla el entorno a través del *SA* especializado en el sensor. Es decir, tiene el conocimiento de lo que ocurre en el entorno, y sabe como actuar sobre él para lograr los objetivos globales. En el ejemplo de las cámaras PTZ podría consistir en mover las cámaras coordinadamente con otros agentes para seguir a un determinado objetivo.

Si bien es cierto que este agente también debe estar especializado para interactuar con su *SA* asociado, este podrá ser diseñado de forma más genérica por cada tipo de sensor o actuador. Siguiendo con el ejemplo de las cámaras PTZ, podemos encontrarnos con diferentes fabricantes que proporcionen un acceso al dispositivo (protocolos, interfaces de conexión, etc)

totalmente diferentes. Esto requerirá por tanto especializar los *SA* para poder interactuar con los dispositivos. Sin embargo las diferentes cámaras posiblemente dispongan de una funcionalidad similar: proporcionar un control para cambiar el campo de visión de la cámara, y generar imágenes en tiempo real. Esto puede ser aprovechado en el desarrollo del sistema multi-agente, permitiendo diseñar un único *CA* especializado en las cámaras PTZ, independientemente del dispositivo concreto que se encuentre por debajo.

Dentro del modelo de fusión JDL, este agente podría enmarcarse dentro de diferentes niveles. Principalmente estaría relacionado con el nivel 4, que es el nivel donde se refina el sistema de fusión para mejorar el proceso o la tarea de vigilancia, que entre otras cosas puede incluir la gestión de los sensores. Sin embargo este agente también deberá integrar lógica de nivel 2 y en algunos casos de nivel 3 si quiere reaccionar o adelantarse a los cambios del entorno. Por lo tanto, este agente dentro del modelo JDL se encargará de evaluar la situación y realizar un proceso de gestión sobre los sensores/actuadores.

El despliegue de estos agentes en la arquitectura dependerá del tipo de sensor que gestione y la tarea global que se esté intentado llevar a cabo. Podríamos encontrarnos *CA* que sólo necesiten interactuar con su sensor asociado, u otros que necesiten colaborar entre ellos para solucionar un problema que de forma independiente no sería viable, agentes que necesiten agentes de fusión para poder tomar decisiones correctas, etc. La cantidad de combinaciones *SA*-*CA*, CA-*CA*, CA-*SA*, *FA*-*CA* que se podrían llegar a presentar serían infinitas, y dependerán principalmente del problema que se esté tratando de resolver.

En las siguientes subsecciones se describen las interacciones más comunes, así como sus posibles casos de uso.

#### 3.3.3.1 Agente de Control mono-sensor

Un *CA* mono-sensor sería aquel que no requiere interactuar con más agentes que con su sensor asociado, a excepción del Agente Operador (que se comentará más adelante) que será el que establezca los objetivos. Con esta información es capaz de satisfacer el objetivo que se le ha asociado. Un ejemplo podría consistir en un termostato, que podría estar representado por un *SA*, ya que este es capaz de interactuar con el entorno midiendo la temperatura y conmutando el encendido la caldera. El *CA* sería el encargado de llevar a cabo la lógica de control propiamente dicha, conociendo la temperatura objetivo y la temperatura actual, por ejemplo implementando un controlador PID. En la figura 3.9 se observan un par de *CA* interactuando exclusivamente con sus sensores asociados.

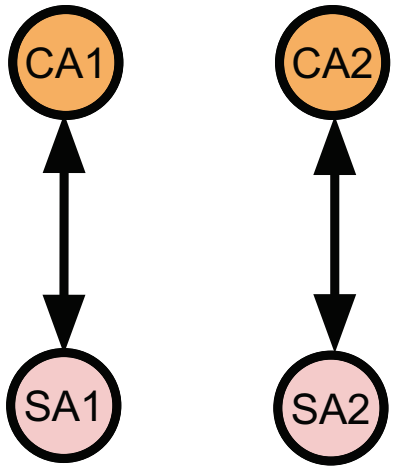


Figura 3.9: Agente de control (*CA*) mono-sensor. Su despliegue no requiere de la interacción con otros agentes homólogos u otras fuentes de información distintas de las proporcionadas por su sensor.

#### 3.3.3.2 Agente de Control multi-sensor

Un *CA* podría estar asociado a diferentes sensores para llevar a cabo su tarea. Por ejemplo, podríamos contar con un sensor de presencia y una cámara que tiene que obtener fotografías cuando se detecte movimiento. En este caso el sensor de presencia notificaría al agente de control de la cámara la presencia de objetivos que fotografiar, y este actúa sobre la información proporcionada por la cámara para guardar las imágenes.

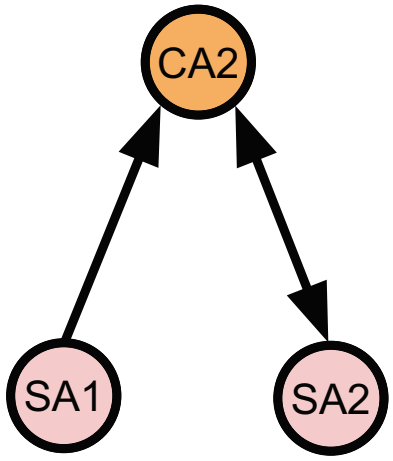


Figura 3.10: Agente de Control (*CA*) multi-sensor que cuenta con la información de un sensor para la gestionar la información generada por otro sensor.

Aunque este agente sea capaz de recibir información del entorno a través de diferentes *SA*, la idea es que un *CA* sea capaz de controlar/manejar un único *SA* de forma simultánea. Esto

es así por simplificar la lógica de control de cada agente, minimizar la información necesaria para operar, y seguir un planteamiento distribuido que no dependa de un único nodo central. Por lo que un *CA* podrá contar con múltiples entradas de información, pero sólo una única salida de control. El esquema de este tipo de agente puede observarse en la figura 3.10. Se puede ver cómo hay dos sensores *SA*1 y *SA*2 proporcionando información, pero hay una única salida de control hacia *SA*2.

#### 3.3.3.3 Agente de Control colaborativo

En este tipo de *CA* hay conciencia de otros *CA* desplegados en el entorno, y podría llegar a ser necesaria una colaboración, ya sea de forma puntual o continuada, para satisfacer un objetivo global determinado por el operador. Por ejemplo, podríamos contar con dos semáforos que controlan el acceso a una vía de un único carril. En este caso contaríamos con dos elementos de control, como son los semáforos, representados por dos *SA*, y dos *CA* actuando sobre cada uno de ellos. Cuando fuera necesario habilitar el paso de vehículos en uno de los dos sentidos, los agentes de control deberían coordinarse para cambiar el estado de los semáforos, uno poniéndolo en rojo, y el otro en verde. Esta disposición de los agentes podría quedar representada por la figura 3.11, donde cada *CA* controla un único *SA*, pero los *CA* colaboran entre sí para satisfacer el objetivo.

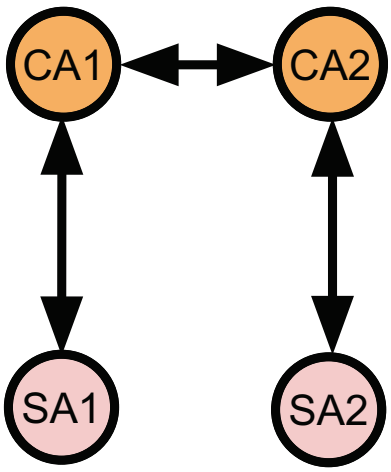


Figura 3.11: Dos Agentes de Control (*CA*) colaborando para actuar sobre dos agentes sensor (*SA*).

Sería posible mejorar el comportamiento de los *CA* descritos en el ejemplo anterior simplemente añadiendo más *SA* que proporcionen información adicional del entorno, como puede ser sensores de presencia de vehículos en cada uno de los sentidos. Estos nuevos sensores se desplegarían como nuevos *SA* que proporcionan más información a sus respectivos *CA*. Cuando un *CA* detecte que hay cierta cantidad de coches esperando, podría proactivamente iniciar una colaboración con su homólogo para cambiar el sentido de la marcha.

#### 3.3.3.4 Agentes de Control colaborativos con información de fusión

En este tipo de interacción contamos con varios *CA* que pueden llegar a necesitar información de fusión para satisfacer de forma conjunta un objetivo de alto nivel, y que de otra forma no serían capaces de solucionar. Por ejemplo, podríamos contar con un conjunto de cámaras PTZ desplegadas por el entorno, y un conjunto de sensores de localización que proporcionan información de posición sobre determinados objetivos del entorno. La tarea podría consistir en la monitorización no redundante de los diferentes objetivos utilizando para ello las cámaras. Sin tener en cuenta la fusión de datos, podría ocurrir que un mismo objetivo sea percibido más de una vez por los sensores de localización. Esto podría ocasionar que dos cámaras estuvieran siguiendo el mismo objetivo pensando que son dos entidades distintas, cuando realmente se trata de la misma.

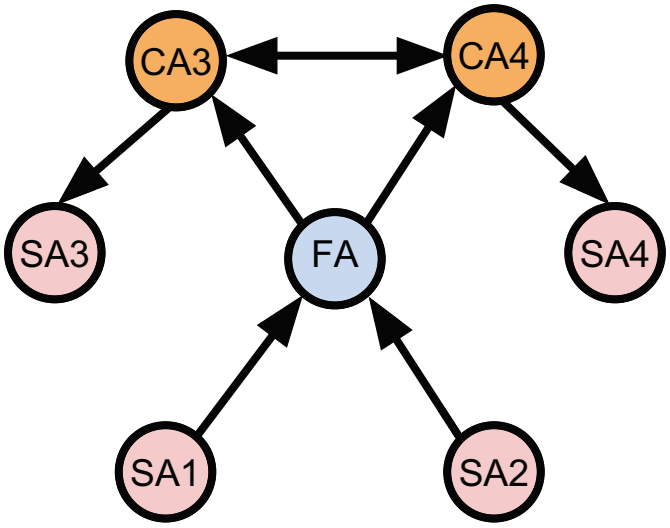


Figura 3.12: Ejemplo de dos Agentes de Control (*CA*3 y *CA*4) utilizando información de un Agente Fusión (*FA*) para controlar dos Agentes Sensor (*SA*3 y *SA*4). La información sobre el entorno estaría generada por los Agentes Sensor *SA*1, y *SA*2, cuya salida podría ser redundante.

En este caso los *CA* gestionando las cámaras necesitan una fuente de información no redundante que les ayude a mejorar la toma de decisiones. Para ello sería posible añadir un Agente de Fusión (*FA*) que integre la información de los diferentes sensores de localización utilizando alguna de las arquitecturas de fusión descritas anteriormente. El despliegue de estos agentes quedaría descrito en la figura 3.12, donde contaríamos con dos agentes cámara (*SA*3 y *SA*4), dos agentes de control de cámara (*CA*3 y *CA*4), dos sensores de localización (*SA*1 y *SA*2), y un agente integrando su información (*FA*). Este despliegue solucionaría el problema mencionado anteriormente, ayudando de forma transparente a los *CA* a llevar a cabo su tarea de manera más eficiente.

Se podrían presentar otros muchos despliegues donde la información de fusión fuese necesaria.

En el ejemplo anterior podríamos prescindir de los sensores de localización y utilizar únicamente la información proporcionada por las cámaras tras realizar un análisis sobre las imágenes. En este caso los sensores que proporcionarían la información del entorno serían los mismos sensores a controlar. Si las cámaras tuviesen puntos de vigilancia en común tendríamos el mismo problema descrito anteriormente. En este caso bastaría con incluir un *FA* a la salida de la información de las cámaras, y utilizar a su vez esta información para gestionarlas, tal y como se muestra en la figura 3.13.

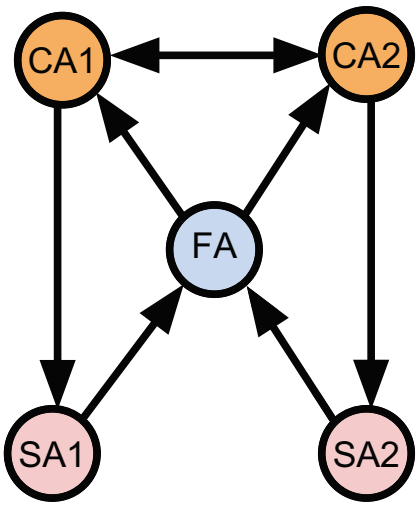


Figura 3.13: Ejemplo de dos Agentes de Control (*CA*1 y *CA*2) utilizando información de un Agente Fusión (*FA*) para controlar dos Agentes Sensor (*SA*1 y *SA*2). La información sobre el entorno estaría generada por los Agentes Sensor *SA*1, y *SA*2, cuya salida podría ser redundante.

Como se puede apreciar, el Agente de Control (*CA*) es la pieza fundamental de la arquitectura que permitiría llevar a cabo las tareas de gestión establecidas en función de la información proporcionada por el entorno. La información del entorno sería percibida a través de un Agente Sensor (*SA*), o un Agente Fusión (*FA*), dependiendo de las características particulares del entorno. Para llevar a cabo estas tareas los agentes podrán colaborar entre sí, si la tarea lo requiere. Podemos resumir la información de entrada y salida de este agente en la tabla 3.3.

| | |
|---|---|
| **Entrada** | |
| **Información de sensor** | El *CA* podrá recibir información directamente de los sensores de nivel 0/1 del modelo JDL a través de uno o varios *SA* que se encuentren desplegados por el entorno. Esta información podría consistir en los diferentes blancos detectados en el entorno, su posición ,etc. |
| **Información de fusión** | De manera adicional, si contamos con información de fusión de nivel 1 a través de algún *FA*, este agente también la podrá recibir. Esta entrada deberá considerarse como la información proporcionada por un sensor más. De tal forma que la integración de la fusión de datos sea transparente para el *CA*. |
| **Coordinación** | Si un *CA* necesita coordinarse con otros agentes, este podrá recibir un flujo de información que represente la información de coordinación necesaria para llevar a cabo una tarea de forma coordinada. |
| **Control** | Un agente de control podrá recibir también información de control. Esta será proporcionada principalmente por los Agentes Operador (*OA*), y consistirá principalmente en la asignación de una tarea o establecimiento de un objetivo de gestión. |
| **Salida** | |
| **Coordinación** | En las tareas que requieran coordinación, este agente podría iniciar la comunicación necesaria para comenzar o gestionar este tipo de procesos. Normalmente el destinatario de este flujo de control, serán otros *CA* homólogos. |
| **Control** | La tarea principal de un *CA* será la gestión y control de un sensor asociado. Ya sea a través de la coordinación o no con otros *CA*. Por lo que este flujo de información representará a la gestión que se realiza sobre el sensor, y en concreto sobre el *SA* que lo represente. |

Tabla 3.3: Información de entrada y salida del Agente Control (*CA*).

### 3.3.4 Agentes de Eventos

Cualquier arquitectura que integre sensores monitorizando un entorno determinado, normalmente requerirá poder almacenar cierta información relativa a su estado, o a los eventos más importantes que hayan ocurrido. Esto suele ser necesario para poder analizar a posteriori cómo ha sido la evolución del entorno, evaluar su correcto funcionamiento, detectar anomalías, reproducir escenarios, etc. Para ello sería necesario incluir otro agente llamado Agente de

Eventos (EA) que se encargue de guardar esta información en el medio más conveniente.

Este agente sería una especie de agente sumidero que permite recibir información de múltiples fuentes conectadas a él, y que dependiendo de como se implemente podría permitir almacenar los datos en un base de datos, en un fichero, subirlos a cualquier tipo de almacenamiento en la nube, etc. Esto permite abstraer al resto de agentes de la arquitectura del medio concreto en el que se está almacenando la información, de cómo acceder a él, evitar problemas de concurrencia de escritura, etc.

Este agente, dentro del modelo de fusión JDL, no se encontraría dentro de ningún nivel específico, pero sí podría considerarse como parte de gestión de bases de datos. Ya que de alguna forma este agente permitiría la interacción con diferentes sistemas que información que podrían servir en un futuro como una fuente de información más, sobre la que se podría razonar y obtener patrones para mejorar los procesos de fusión.

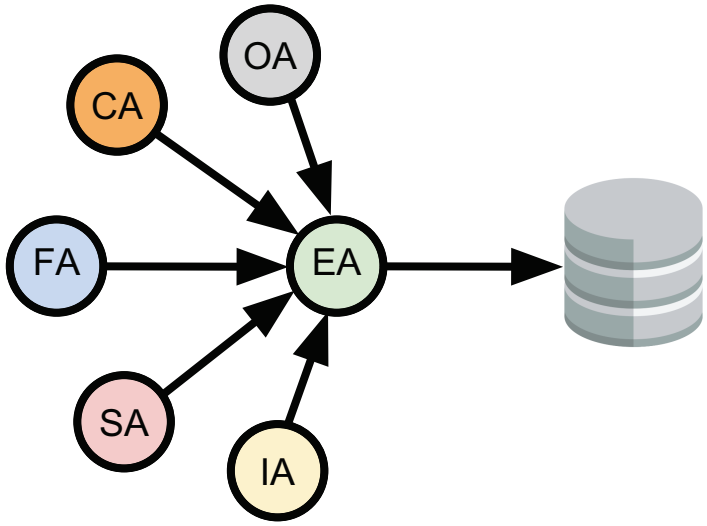


Figura 3.14: Ejemplo de Agente de Eventos (EA) almacenando eventos producidos por diferentes agentes de la arquitectura en una base de datos. El resto de agentes no tienen por qué conocer el medio donde se almacena la información, permitiendo realizar cambios en el sistema de almacenamiento de manera transparente.

En la figura 3.14 se muestra un diagrama donde agentes de diferentes tipos (*IA*, *CA*, *FA*, y *SA*) se encuentran enviando información a un EA que estaría almacenando la información en una base de datos. Un EA no tiene por qué ser único dentro de la arquitectura, ya que nos podría llegar a interesar guardar la información en diferentes medios o formatos, que sean más apropiados dependiendo de los datos que estemos manejando.

Por ejemplo, podríamos contar con un EA especializado en grabar las secuencias de vídeo generadas por los *SA* en archivos de vídeo. O podríamos contar con un EA especializado en almacenar en una base de datos los eventos producidos por el sistema, como pueden ser intrusiones en determinadas zonas, detecciones de movimiento, etc. Lo que si se debería

cumplir es la premisa de que un EA sólo puede recibir información de un determinado tipo para almacenarla/transmitirla a un determinado medio. Esto permitiría desarrollar múltiples EA que pueden ser fácilmente desplegados en la arquitectura sin tener que complicar la lógica del agente en exceso (igual que ocurría con el *CA*), además de distribuir el trabajo entre varios agentes en caso de que el volumen de información sea demasiado alto.

Una aplicación que podría resultar de gran utilidad, sería la creación de un *EA* que transmitiese determinada información del entorno a sistemas de computación en la nube. Los entornos multi-sensor a menudo generan una gran cantidad de información, que aparte de ser gestionada en tiempo real, podría utilizarse además para determinados análisis como la detección de patrones, minería de datos, identificar asociaciones, etc. Esto es conocido como *Big Data* o datos masivos. Por lo que sería posible crear un *EA* especializado en la comunicación con otros procesos distribuidos que realizasen estas tareas. En la figura 3.15 se puede observar una representación de este agente que estaría recibiendo la información de un subconjunto de agentes de la arquitectura .

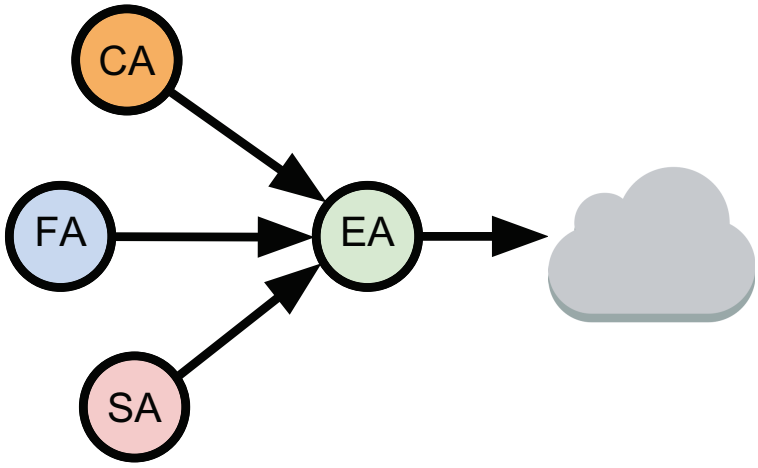


Figura 3.15: Ejemplo de Agente de Eventos (EA) recibiendo información de un subconjunto de agentes para su transmisión a sistemas de computación en la nube. Esto podría resultar de gran utilidad para aplicar técnicas Big Data.

Debido a la cantidad de agentes que podrían estar involucrados en la interacción con el EA, y por simplificar su integración en la arquitectura, este agente debería ser capaz de ser descubierto automáticamente por el resto de agentes. Es decir, cuando un agente sea capaz de generar información de un determinado tipo, susceptible de ser almacenada, preguntará al resto de agentes si hay alguno que sea capaz de realizar esta tarea. Si se encontrase algún EA desplegado en la arquitectura con estas capacidades, entonces automáticamente se formaría un vínculo de conexión entre los dos agentes. De esta formar sería muy sencillo agregar o quitar capacidades de registro de datos, ya que sería una tarea totalmente transparente para el resto de agentes.

La entrada y salida de información de este agente queda descrita en la tabla 3.4.

| | |
|---|---|
| **Entrada** | |
| **Eventos/Información** | Se trata de los eventos o información generada por otros agentes de la arquitectura, y que normalmente será información útil para su monitorización a posteriori. La información recibida por este agente deberá ser de un único tipo, ya sean eventos de fusión, de detección de presencia, secuencias de vídeo, etc. Esto permitirá especializar al agente en el tratamiento de estos datos, por ejemplo aplicando filtros, convirtiéndolos a otros formatos, etc. |
| **Salida** | |
| **Almacenamiento/Transmisión** | La información recibida podrá ser almacenada o transmitida a diferentes medios, como ficheros, bases de datos, servicios en la nube, etc. La información de salida podría estar modificada o filtrada con respecto a la información de entrada, por ejemplo tras la conversión a un formato determinado. El agente debería ser capaz de transmitir la información únicamente a un tipo de medio en concreto. |

Tabla 3.4: Información de entrada y salida del Agente de Eventos (EA).

### 3.3.5 Agente Operador

El Agente Operador (*OA*) tiene por misión representar al operador real, que es aquel que indicará el objetivo a seguir por la arquitectura multi-agente. Como un objetivo determinado normalmente tendrá que ser acometido entre múltiples agentes desplegados en la arquitectura, no tiene sentido que un operador de alto nivel tenga que conocer cada uno de los *SA*, o *CA* en ejecución para asignarles la tarea. Es el *OA* el que conoce el despliegue del sistema multi-agente, su estado, define las tareas que son posibles llevar cabo, y sabe como indicar al resto de agentes las tareas independientes a realizar para satisfacer los objetivos establecidos. De esta forma, un operador real que quiera establecer una serie de objetivos determinados (normalmente a través del Agente Interfaz descrito a continuación) simplemente tendrá que indicárselo al *OA* correspondiente.

Un *OA* normalmente se comunicará con los agentes *CA* encargados de actuar sobre el entorno y con el Agente Interfaz (*IA*), descrito más adelante, encargado de atender las peticiones del operador final. Esta relación se muestra en la figura 3.16. En principio podría haber diferentes

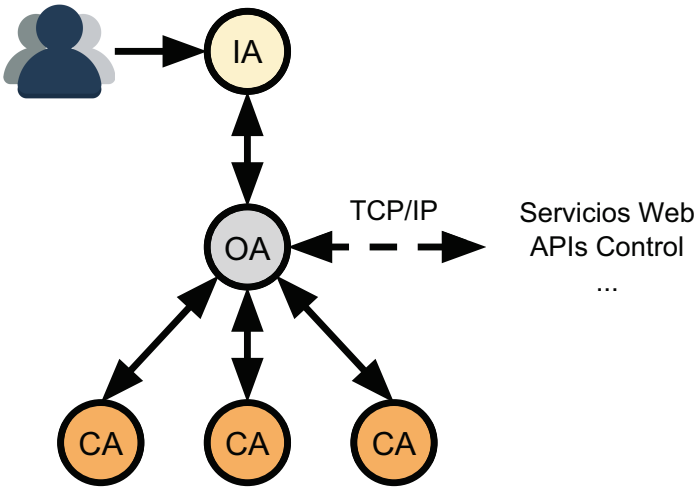


Figura 3.16: Agente Operador (*OA*) gestionando un conjunto de Agentes de Control (*CA*). Esta gestión se realiza en función de las tareas establecidas por los operadores a través del Agente Interfaz (*IA*), o aplicaciones remotas como servicios web.

*OA* desplegados en el sistema que atiendan a diferentes objetivos y tareas, pero siempre y cuando los objetivos establecidos en diferentes *OA* no entren en conflicto entre sí. Es decir, cada *OA* finalmente gestionará un conjunto de *CA* que realizarán las tareas que se les indiquen, y no sería recomendable que *OAs* diferentes gestionen los mismos *CAs* que pudieran estar recibiendo asignaciones de tareas diferentes.

Además de la interacción a través del *IA*, podría ser interesante que este agente además pudiese contar con una serie de interfaces de gestión externas al sistema multi-agente, y que permitiese a determinados procesos remotos iniciar o parar los objetivos. De este modo sería posible configurar el sistema multi-agente a través de interfaces web, u otro tipo de conexiones a través de Internet. Para ello sería posible contar con una interfaz de comunicaciones a través de un protocolo de transporte de red como TCP/IP, y que implemente la funcionalidad adecuada en forma de APIs para iniciar los servicios disponibles. Esto le daría más versatilidad al sistema multi-agente desarrollado, permitiendo su gestión, por ejemplo, a través de teléfonos móviles.

Si observamos el modelo JDL, este también tiene en cuenta la interacción con el operador a cargo del sistema de fusión. En cierto modo, este agente, junto al Agente Interfaz que se describe a continuación, se encontrarían a este nivel. De este modo, estos serían agentes estarían directamente relacionados con el operador y lo representan de algún modo dentro de la arquitectura multi-agente.

Los diferentes elementos de entrada y salida de información del *OA* se definen en la tabla 3.5.

| | |
|---|---|
| **Entrada** | |
| **Objetivos** | Cada *OA* integra un conjunto de objetivos que pueden establecerse dependiendo del entorno que estemos tratando. Por este motivo, la entrada principal de este agente consistirá en los comandos de control necesarios para iniciar o detener los objetivos. |
| **Salida** | |
| **Control** | Dependiendo del comando de control suministrado a la entrada, este agente deberá proporcionar una salida de control a los *CA* correspondientes necesarios para llevar a cabo la tarea u objetivo. |
| **Tareas** | Es necesario que otros agentes conozcan las tareas disponibles a realizar, por ejemplo para presentarlas en la interfaz de operador. Por este motivo, cada *OA* deberá ser capaz de proporcionar esta información a otros agentes que lo necesiten. |

Tabla 3.5: Información de entrada y salida del Agente Operador (*OA*).

### 3.3.6 Agente Interfaz

Un Agente Interfaz (*IA*) tendrá la habilidad de generar una interfaz hombre - máquina (HMI). Esta interfaz es el medio por el que se presentan los datos o eventos más importantes a un operador (humano) y a través del cual éste controla el proceso. Este agente podría llegar a representar lo que industrialmente se conoce como SCADA (Supervisión, Control y Adquisición de Datos). Esta parte de la arquitectura es de vital importancia ya que es desde donde se puede observar el correcto funcionamiento de todo el sistema, establecer tareas, visualizar alarmas, y en general presentar cualquier información que vaya a ser relevante para el operador.

La información que se suele mostrar al operador en entornos donde entran en juego sensores suele consistir principalmente en la información muestreada. Como puede ser las imágenes de las cámaras, las detecciones de las diferentes entidades sobre un mapa, el estado de una cadena de procesos, temperaturas, presiones, etc. Esto dependerá principalmente del dominio sobre el que estemos trabajando.

Al igual que el *OA*, este agente está directamente relacionado con el operador humano a cargo del sistema, por lo que estaría dentro del modelo JDL en el nivel de interacción. De esta forma, es como el operador puede observar el estado del entorno, tomar decisiones, y actuar sobre la arquitectura multi-agente para establecer diferentes objetivos o tareas.

La interacción de este agente con otros agentes de la arquitectura se describe en la figura 3.17. En esta se puede observar un *IA* recibiendo información de otros *SA* que se encuentran

monitorizando el entorno. Esta información sería presentada en tiempo real al operador a través una interfaz de usuario en un ordenador. En esta figura también se puede apreciar como el agente *IA* interacciona con un Agente Operador (*OA*), que permite interactuar con el resto de la arquitectura multi-agente para el establecimiento de tareas o comportamientos.

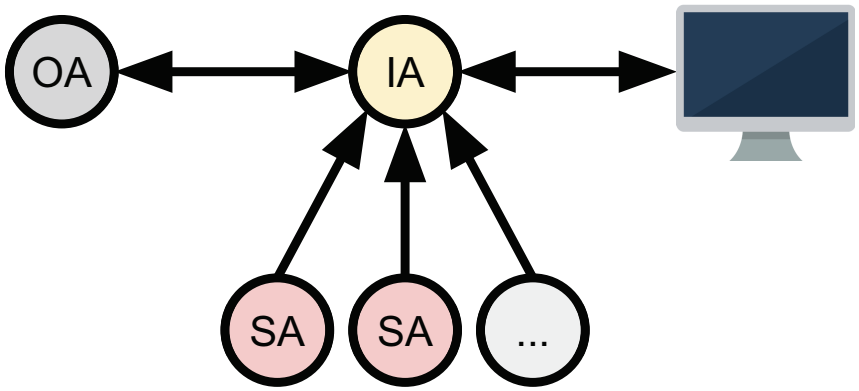


Figura 3.17: Agente Interfaz (*IA*) mostrando la información proporcionada por diferentes *SA*. El *IA* además podrá generar eventos de control hacia los *OA* asociados dependiendo de su interacción con el operador.

La información recibida del resto de agentes no tiene por qué limitarse a la información percibida únicamente por los sensores. Cualquier agente de la arquitectura podría generar información para su monitorización. Por ejemplo, si contamos con algún Agente Fusión capaz de suministrar información más refinada sobre las entidades del entorno, tendría más sentido poder monitorizar esta que no la información suministrada directamente por los sensores. O si contamos con un agente que genera alarmas, ya sea un *SA*, o un *CA*, dependiendo de determinadas condiciones del entorno, es obvio que también sería interesante poder monitorizarlas. La entrada y salida de este agente quedaría reflejada en la tabla 3.6.

| | |
|---|---|
| **Entrada** | |
| **Información del entorno** | El *IA* recibirá principalmente información del entorno a través otros agentes desplegados, ya sean Agentes Sensor, Agentes de Control, Agentes de Fusión, etc. Esta información podrá consistir en información sin procesar (la que genera directamente el sensor), o procesada/generada por cualquier otro agente. |
| **Interacción con el operador** | El operador podrá gestionar los diferentes objetivos de gestión disponibles en los *OA* desplegados en el sistema. Esto se transformará en una interacción sobre la interfaz proporcionada por el agente interfaz. |
| **Salida** | |
| **Objetivos** | El operador podrá generar eventos de control a través de la interfaz gráfica. Estos eventos se propagarán a los diferentes *OA* que sean necesarios para llevar a cabo los objetivos seleccionados. |
| **Visualización** | Este agente consistirá principalmente en una interfaz gráfica que proporciona información valiosa para el operador. Su salida por tanto se compondrá de los diferentes datos u eventos recibidos a través del resto de agentes. Esta información podría contener imágenes de vídeo, datos en tiempo real de sensores de medición, estados de los autómatas, etc. |

Tabla 3.6: Información de entrada y salida del Agente Interfaz (*IA*).

## 3.4 Conclusiones

En esta sección se ha tratado con el problema de gestión multi-sensor y se ha realizado el diseño de una arquitectura multi-agente para su gestión. Esta arquitectura se ha definido a nivel del dominio del problema, definiendo las diferentes entidades en términos de agentes, la información que gestionan, y la relación que tienen entre ellos dentro de la arquitectura.

En este diseño se han contemplado diferentes agentes que se enmarcan dentro del modelo de fusión JDL. EL agente de más bajo nivel, el Agente Sensor (nivel 0/1), está relacionado con el procesamiento de la información del sensor y su integración dentro del sistema multi-agente. También se contempla el Agente Fusión (nivel 1), que sería el encargado de integrar la información generada por múltiples sensores. Esta información puede ser aprovechada por los Agentes de Control (nivel 2, 3, y 4), que evaluarán la situación del entorno para realizar la gestión específica sobre cada sensor particular para cubrir las tareas especificadas. Además se contemplan otros agentes relacionados con la gestión de información, como el Agente de Eventos, y la inclusión del operador en la arquitectura a través del Agente Operador y el Agente Interfaz.

Este diseño basado en sistemas muti-agente permite el despliegue de diferentes entidades autónomas distribuidas que pueden ser organizadas de diferentes formas dependiendo de la tarea a satisfacer. Por ejemplo, la fusión de datos se puede contemplar como una arquitectura centralizada, jerárquica, o distribuida. Esto dependerá del entorno y sus propiedades. De igual modo se han considerado diferentes aplicaciones del Agente Control, que permite la gestión de los sensores o actuadores en base a información generada por el propio sensor, por otro sensor, o por el resultado de la fusión. De este modo podemos cubrir un gran número de casos de uso gracias a la flexibilidad en el diseño de entidades autónomas en forma de agentes.

Por lo tanto, esta arquitectura podrá ser aplicada en diferentes entornos de gestión multi-sensor, y en cada una de estas aplicaciones habrá que seleccionar y especificar los agentes necesarios para satisfacer determinadas tareas. Esta especificación tendrá en cuenta la selección específica de los sensores, la información que estos son capaces de generar, la fusión de datos que se puede aplicar sobre estos datos, así como definir las tareas de control y coordinación que deberían realizar los agentes.

De este modo, en las secciones 4 y 5 se describirá la instanciación específica de este diseño sobre dos entornos de vigilancia diferentes. Uno aplicado sobre un entorno multi-cámara, y otro sobre un entorno de vigilancia marítimo. De esta forma podremos comprobar como adecuar este diseño genérico y así evaluar su adecuación y correcto funcionamiento.

# 4

# Entorno Multi-Cámara

## 4.1 Introducción

Hoy en día los sistemas de vigilancia están evolucionando hacia complejos sistemas de información que son capaces de proporcionar grandes cantidades de información acerca de nuestro entorno, recogida a través de redes de sensores espacialmente distribuidas. Los avances en las tecnologías sobre las que se apoyan como la comunicación digital, la transmisión de vídeo, y en especial las redes de sensores y actuadores inalámbricas (WSANs) (Akyildiz and Kasimoglu, 2004) han facilitado el despliegue de gran cantidad de sensores utilizados en monitorización ambiental, sistemas de salud, seguridad del hogar, seguridad pública, y en general en entornos críticos.

Las cámaras de vídeo tal vez sean el sensor más importante en los entornos de vigilancia, ya que desde siempre ha sido uno de los sensores más investigados por la comunidad científica. Hoy día existen redes de sensores de visión que permiten no solo proporcionar imágenes sobre el entorno, si no que tienen capacidades adicionales como la detección y seguimiento de objetos, clasificación de colores, reconocimiento de actividad, detección de intrusiones, detección de actividades sospechosas, y muchas otras características (Hampapur, 2008). Estas novedosas capacidades son normalmente incluidas en los sistemas de vigilancia tradicionales, lo que mejora el rendimiento en la tarea de vigilancia (Han and Bhanu, 2007).

En esta sección nos vamos a centrar por tanto en el uso de las cámaras como sensor de visión, y su integración en un sistema multi-agente real. En concreto contaremos con cámaras pan-tilt-zoom (PTZ). Este tipo de sensores, a diferencia de los sensores de visión tradicionales, tienen la capacidad de cambiar su campo de visión según sea necesario. Un operador de este tipo de cámaras normalmente las controlaría mediante el uso de alguna palanca de mando (o joystick) según las percepciones que la propia cámara le proporciona y los objetos de interés que esté deseando seguir. Esto les da cierta flexibilidad, ya que no se encuentran atados a las percepciones proporcionadas por una cámara fija.

Sin embargo el uso de estas cámaras requiere de más atención por parte de los operadores, ya que controlarlas supone prestar atención completa al movimiento de la cámara. Sería difícil que un operador pudiese gestionar más de una cámara de forma simultánea de una manera adecuada. Esto supone un problema cuando nos encontramos con un gran entorno de vigilancia con un gran número de estas cámaras. Tendría sentido por tanto el uso de algún tipo de sistema autónomo capaz de controlar automáticamente las cámaras en función de determinados objetivos establecidos por el operador. El operador en este caso se podría limitar a visualizar las imágenes y establecer eventualmente algún objetivo de monitorización.

Hay dos problemas fundamentales a solucionar a la hora de diseñar este tipo de sistema de visión autónomo, tal y como se describe en (Manyika and Durrant-Whyte, 1994). El primer problema es la fusión de información, que como hemos visto en el estado del arte de la tesis consiste en la combinación de información de diferentes fuentes (Liggins II et al., 2008). El segundo problema consistiría en la gestión multi-sensor, que asume que el problema de la fusión está solucionado, y que debería optimizar el funcionamiento global del sistema aplicando operaciones individuales en cada sensor (Molina López et al., 2003). Los dos enfoques principales para llevarlo acabo sería mediante arquitecturas centralizadas o descentralizadas. En una arquitectura centralizada normalmente nos encontraríamos con un centro de fusión o un nodo central de la red, combinando toda la información y planificando la ejecución de las acciones en cada sensor. Este tipo de arquitectura es más sencillo de desarrollar pero tendría serios problemas de escalabilidad en entornos medianamente grandes, baja tolerancia a fallos, y problemas de despliegue cuando los sensores se encuentran altamente distribuidos (Molina López et al., 2003).

Tal y como hemos descrito en la sección 3, nuestro enfoque se basará en la arquitectura multi-agente diseñada, que nos permitirá aplicar un sistema de gestión de forma distribuida. Por lo tanto, en esta sección vamos a especificar el diseño concreto del sistema multi-agente para este entorno. En la sección 4.2 se describe el entorno y los objetivos a cubrir, en la sección 4.3 se propone un diseño de sistema multi-agente para llevar a cabo los objetivos. En la sección 4.4 se describe el proceso de optimización que hemos realizado en la integración de las cámaras en la arquitectura multi-agente, y finalmente en la sección 4.5 se describen algunos de los experimentos realizados.

## 4.2 Descripción del entorno y objetivos

Como se comenta en la introducción, en esta propuesta trabajaremos con un conjunto de cámaras PTZ gestionadas mediante un sistema multi-agente como el descrito en la sección 3. En este caso trabajaremos con un entorno real que nos servirá para afrontar dos objetivos diferentes. Por un lado validar la arquitectura multi-agente descrita en la sección anterior para su aplicación sobre un entorno concreto de vigilancia. Y por otro lado enfrentarnos y resolver los problemas que podemos encontrarnos durante su desarrollo. La motivación principal de querer trabajar con un entorno real es que muchas veces, cuando se diseñan arquitecturas de alto nivel, como la propuesta en esta tesis, se suelen obviar muchos detalles de implementación que no son fáciles de abordar. Por ejemplo, es relativamente fácil definir un agente de tipo cámara con ciertas capacidades que permitan controlar la cámara en función de las percepciones recibidas. Sin embargo, en el entorno real esto suele suponer tener que acceder y procesar las imágenes generadas por la cámara, acceder a sus controles mediante determinados protocolos, transmitir las imágenes, etc. Estas tareas tan básicas necesarias para el supuesto agente sensor no son tan fáciles de conseguir en un entorno real, por lo que merece la pena su estudio. De hecho, el enfrentarnos a un entorno real nos permitió explorar diferentes temas de investigación que se recogen en esta tesis, en concreto el descrito en la sección 4.4, y que inicialmente no se habían contemplado. Este tema en particular dio lugar a una patente.

Además, el uso de la información proporcionada por entornos reales siempre dista mucho de las simulaciones de entornos ideales con los que se suelen probar los diseños. En el mundo real hay muchos aspectos que pueden afectar a este tipo de sistemas que emplean sensores de visión, especialmente las condiciones de iluminación, sombras, oclusiones, etc. Por este motivo, trabajar con en estos entornos supone un grado de incertidumbre aún mayor.

En nuestro caso contamos con un laboratorio de investigación que dispone del equipamiento necesario para trabajar con las cámaras PTZ, y nos permitirá desplegar un sistema multi-agente distribuido. Por un lado contamos con un conjunto de cámaras PTZ situadas en el laboratorio tal y como se muestran en la figura 4.1. Por otro lado contamos con un conjunto de servidores donde se encuentran conectadas las cámaras. Las cámaras en este caso no proporcionan una interfaz de video digital, por lo que se encuentran conectadas a unas digitalizadoras de vídeo de la marca Matrox. El control de las cámaras se realiza mediante un protocolo de control llamado VISCA que funciona a través del puerto serie de los servidores.

Sobre este entorno queremos diseñar un sistema, siguiendo la arquitectura multi-agente propuesta, para realizar un control autónomo de las cámaras. Por control autónomo nos referimos a añadir agentes al entorno que sean capaces de controlar la orientación de las cámaras automáticamente para seguir determinados elementos del entorno. Por simplificar el problema, el sistema multi-agente se centrará en seguir automáticamente objetivos de un determinado color, pero sería fácilmente extrapolable a problemas más complejos que necesiten

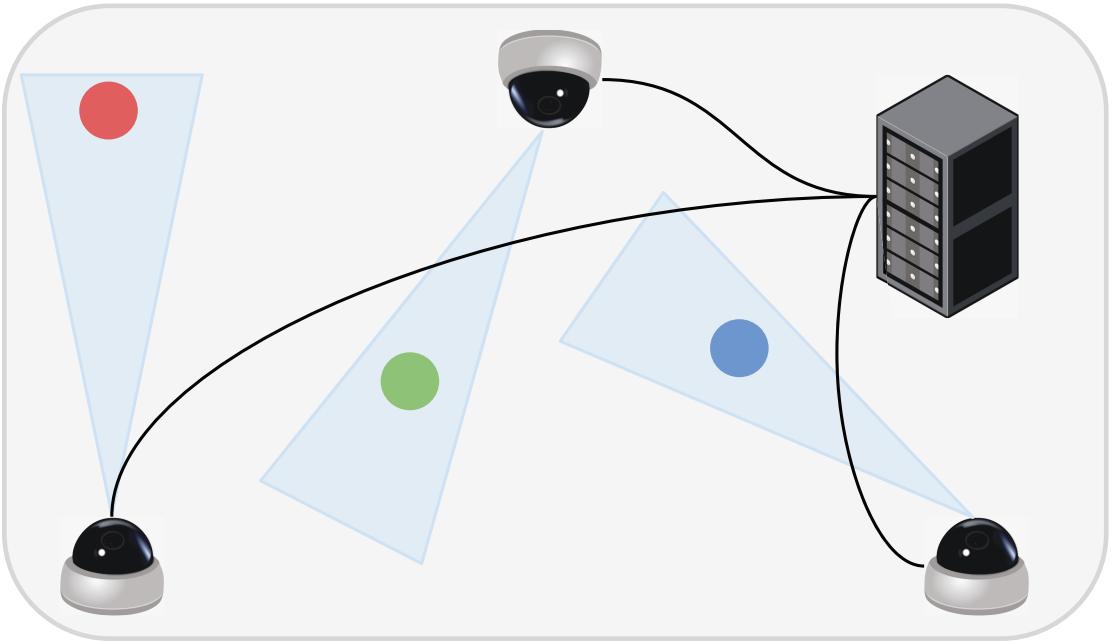

Figura 4.1: Entorno de pruebas sobre el que se planteará el diseño del sistema multi-agente. Este cuenta con tres cámaras PTZ, y un armario de servidores donde están conectadas las cámaras. Desde estos servidores se tiene acceso tanto al vídeo como al control de la cámara. Cada cámara se encuentra conectada a un servidor diferente.

seguir objetos de una determinada forma (como personas), que estén realizando algún tipo de actividad en concreto, que vayan a una determinada velocidad, que pasen por una determinada zona, etc.

Este objetivo de alto nivel nos permitirá evaluar la arquitectura multi-agente a diferentes niveles, desde los Agentes Sensor (*CA*) representando a las cámaras, los Agentes de Control (*CA*) gestionando las cámaras, los Agentes Fusión (*FA*) fusionando la información de las diferentes cámaras, y el Agente Interfaz (*IA*) permitiendo la visualización del entorno al operador.

## 4.3 Diseño del sistema multi-agente

En esta sección se describe la integración de la arquitectura propuesta en la sección 3 para el entorno real descrito anteriormente. Para ello definiremos la adaptación de cada tipo de agente al entorno específico con el que estamos trabajando, la videovigilancia con un sistema puramente multi-cámara. Para ello necesitaremos utilizar gran parte de los agentes definidos en la arquitectura general.

Podemos considerar un conjunto de agentes como los mostrados en la figura 4.2. Al tener que gestionar tres cámaras diferentes, contaremos con tres Agentes Sensor (*SA*) que son los responsables de representar a los sensores de visión dentro de la arquitectura. Estos agentes serán capaces de proporcionar determinada información, como la secuencia de imágenes para su visualización por parte del operador, así como información ya procesada, que consistiría en un conjunto de detecciones o pistas. Debido a la disposición del escenario con el que estamos tratando, como se puede ver en la figura 4.1, es posible que las cámaras puedan llegar a proporcionar información redundante. Es necesario por tanto que la información de las pistas proporcionadas por los sensores pasen por un Agente Fusión (*FA*) que se encargue de integrar toda la información. La salida de información ya fusionada alimentará a los Agentes de Control (*CA*) que se encargarán de dirigir las cámaras para que sigan a los objetivos de color presentes en el entorno. Los *CA* estarán comandados por el Agente Operador (*OA*) que recibe los objetivos a establecer a través del Agente Interfaz (*IA*) que interactúa con el operador. En este caso el *OA*, y por tanto los *CA*, sólo serán capaces de llevar a cabo una tarea, que consiste en el seguimiento de objetos de color.

En la figura 4.2 además se muestra una posible distribución de los diferentes agentes en los servidores disponibles. Teniendo en cuenta que cada cámara está conectada a un único servidor, tiene sentido distribuir cada *SA* y *CA* en un mismo servidor. Uno de los servidores contaría además con el *FA*, y el *OA*. Por último, podemos encontrar el *IA* desplegado en otra instancia distinta, que probablemente representaría el puesto del operador. Esta distribución podría ser completamente diferente dependiendo de las necesidades específicas, pero aprovechamos la disponibilidad de diferentes servidores para desplegar un sistema distribuido real. La ventaja de utilizar un sistema multi-agente, y generalmente un entorno de desarrollo que los soporte, es que es posible ejecutar los agentes en cualquier máquina, ya que estos seguirían conservando la capacidad de comunicarse.

A continuación se describen con más detalle los diferentes agentes de la arquitectura multi-agente para el control multi-cámara.

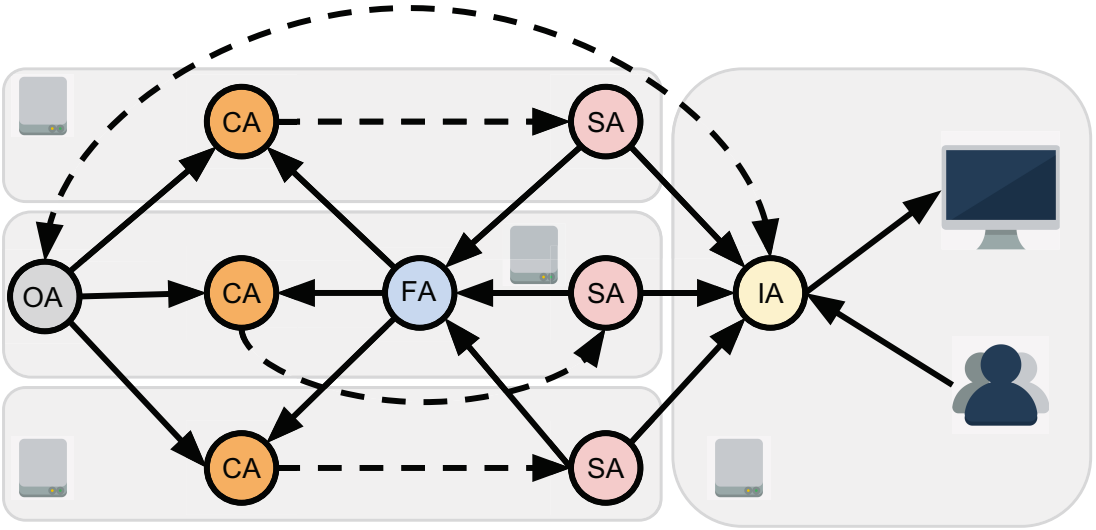


Figura 4.2: Despliegue de arquitectura multi-agente para el control de tres cámaras PTZ. Podemos encontrar tres Agentes Sensor (*SA*) representando las cámaras. Un Agente Fusión (*FA*) que combina la información de las tres cámaras. Tres Agentes de Control (*CA*) que controlan las cámaras. Un Agente Interfaz (*IA*) que interactúa con el operador, y finalmente un único Agente Operador (*OA*) que comanda los objetivos establecidos por el operador. Este despliegue se realizará en diferentes máquinas, aprovechando los servidores disponibles.

### 4.3.1 Agente Sensor

El Agente Sensor (*SA*), para este entorno particular, está directamente relacionado con el sensor de visión. Básicamente proporciona una interfaz de control, en forma de capacidades, que permiten a otros agentes o sistemas interactuar con la cámara para cambiar su campo de visión. Además este agente tiene que tener la capacidad de transmitir las imágenes de la cámara en tiempo real para que un eventual operador pueda monitorizar el entorno.

Para esta tarea, el agente no implementará toda la lógica de control de la cámara (como puede ser implementar el protocolo VISCA), ni gestionará la captura y transmisión de las imágenes en tiempo real (usando tarjetas digitalizadoras). Un *SA* debería ser un componente software ligero, con la capacidad de adaptarse a otros entornos, diferentes tipos de cámaras, etc. Por este motivo, este agente se diseña como un componente que permite integrar otro componente de software de más bajo nivel dentro de la arquitectura multi-agente.

Este componente software ha sido diseñado específicamente para cubrir esta tarea, y la denominaremos como el Controlador del Sensor. Este componente software se describe en la siguiente subsección.

#### 4.3.1.1 Controlador de Sensor

Este controlador será el encargado de manejar el acceso físico a la cámara como tal. Debe ir desplegado en el servidor donde se encuentren conectada la cámara, ya que utilizará determinados recursos hardware para conectarse ella. Estos recursos consistirán en la tarjeta digitalizadora y el puerto de serie. Este componente se construye encima de estos recursos, y permitiría un acceso genérico de los *SA* a las cámaras. Como describimos en (Bustamante and Molina, 2011), actualmente proporciona tres interfaces de alto nivel, independientemente del modelo de cámara que vayamos a utilizar, las cuales se definen a continuación:

**Interfaz de vídeo** Esta interfaz permitirá la transmisión de vídeo en tiempo real a través de la red, utilizando para ello el códec JPEG-2000. Esta interfaz tiene suma importancia dentro de la propuesta de este diseño, ya que al estar tratando con un entorno real de videovigilancia, es completamente necesario cumplir las restricciones de tiempo real impuestas en estos entornos. No sería admisible integrar un sistema autónomo de videovigilancia si esto supone grandes retrasos en el procesamiento o visualización de la información. Por ejemplo, podríamos considerar la transmisión de vídeo en tiempo real a través de un protocolo estándar de comunicación entre agentes, como puede ser el lenguaje FIPA-ACL, disponible en múltiples entornos de desarrollo de agentes como JADE o JADEX. Sin embargo estos protocolos no están diseñados para soportar el volumen de información generado por un sensor de visión, por lo que podríamos saturar el sistema de comunicación de los agentes, añadir grandes retardos, y en definitiva obtener un sistema inutilizable.

Como se describe con mayor detalle en la sección 4.4, este controlador transmitirá las secuencias de vídeo utilizando un protocolo de transporte en tiempo real (Real-Time Transport Protocol - RTP) que permite la codificación y la compresión de fotogramas de vídeo utilizando el estándar de compresión JPEG-2000, directamente soportado por las tarjetas disponibles en nuestro entorno real. Este protocolo además soporta multi-difusión, lo que nos permitirá transmitir vídeo en tiempo real a múltiples destinos de forma simultánea. Esto cumpliría la premisa de que la información de un agente sensor debería poder ser transmitida a múltiples agentes.

**Interfaz de control** La interfaz de control permite ajustar los parámetros de orientación de una cámara PTZ. Estos parámetros incluyen el manejo del movimiento horizontal, vertical y el zoom. Esto permitiría a entidades externas a la cámara, como el *SA*, o un operador, cambiar el campo de visión de la cámara. Es posible que cada modelo de cámara tenga un protocolo de comunicación diferente. En nuestro entorno real es el protocolo VISCA a partir del puerto serie. Cámaras de otros fabricantes podrían proporcionar acceso a través de comunicación TCP/IP. Es por esto importante que esta interfaz sea genérica y

abstraiga al agente del modelo concreto de cámara con el que estamos trabajando. De esta manera podemos reutilizar los *SA* especializados en el control de cámaras PTZ sin tener que adaptarlos a cada modelo o fabricante. El único componente que habría que adaptar sería esta interfaz.

**Interfaz de seguimiento** Esta interfaz probablemente sea la más importante del módulo Controlador del Sensor. Está especializada en ofrecer capacidades de seguimiento a las capas de control superiores. Con capacidades de seguimiento nos referimos a la capacidad de ofrecer información de las pistas que tenemos en el entorno, como el número, la forma, la posición, el color, y demás características que podrían obtenerse del uso de algoritmos de visión. Para este escenario real, como se ha comentado en los objetivos, se proporcionará información de seguimiento basada en determinados colores, pero podría incorporar diferentes tipos de seguimiento dependiendo del entorno con el que trabajemos.

Esta interfaz, al igual que la de control, tiene dos objetivos fundamentales. El primero es abstraer a los agentes de control superiores del algoritmo o sistema de seguimiento que estemos empleando. Los agentes recibirán información sobre las entidades detectadas así como un conjunto de atributos. Esto es extrapolable a cualquier otro algoritmo. El segundo objetivo es evitar la transmisión de vídeo desde el punto donde se captura, hasta el agente que necesita la información.

Si cada agente tuviese que recibir un flujo de vídeo, y procesar las imágenes para generar la información de las pistas, se perdería tiempo en la transmisión y aumentaría el ancho de banda necesario, lo que podría complicar el control en tiempo real de las cámaras. De esta forma, es el propio sensor, a través del *SA*, el que es capaz de procesar las imágenes y generar información de las pistas. Y esta información, mucho más ligera que las secuencias de vídeo ya puede ser utilizada por cualquier agente.

La información que generaría esta interfaz, descrita en forma de eventos, se proporciona en la tabla 4.1. En esta se describen eventos típicos generados por un algoritmo de seguimiento capaz de inicializar, actualizar y borrar pistas en función de la información monitorizada por el sensor. Esta misma información sería retransmitida por el *SA* hacia el resto de agentes que pudieran necesitar la información. Por ejemplo, a un *CA* para seguir a los elementos detectados, un EA para guardar el recorrido realizado por las diferentes pistas, un *IA* para visualizar las pistas detectadas en el monitor, etc.

Este controlador debe estar integrado en el mismo lugar donde vaya a estar conectada la cámara. Hará uso directo de las tarjetas digitalizadoras para el acceso a las imágenes. Utilizará la compresión por hardware para la compresión de imágenes. Accederá al puerto de serie para controlar el movimiento. Y por último, sin necesidad de transmitir la imagen a ningún punto

| Eventos | |
|---|---|
| **Inicialización de pista** | Enviado cuando una nueva pista o entidad ha sido detectada dentro del campo de visión de la cámara, y no ha podido ser asociada con ninguna pista existente. |
| **Actualización de pista** | Enviado cuando una nueva pista ha sido actualizada, por ejemplo, cuando la posición o alguna característica se ha actualizado. La actualización de las pistas se enviarán siempre después de un evento de creado y antes de un evento de borrado. |
| **Borrado de pista** | Enviado cuando una pista inicializada no aparece más en el campo de visión de la cámara. Esto puede suceder cuando una pista lleve un periodo de tiempo determinado sin actualizarse. |

Tabla 4.1: Eventos de seguimiento generados por el *SA* a través de la información recibida del Controlador del Sensor.

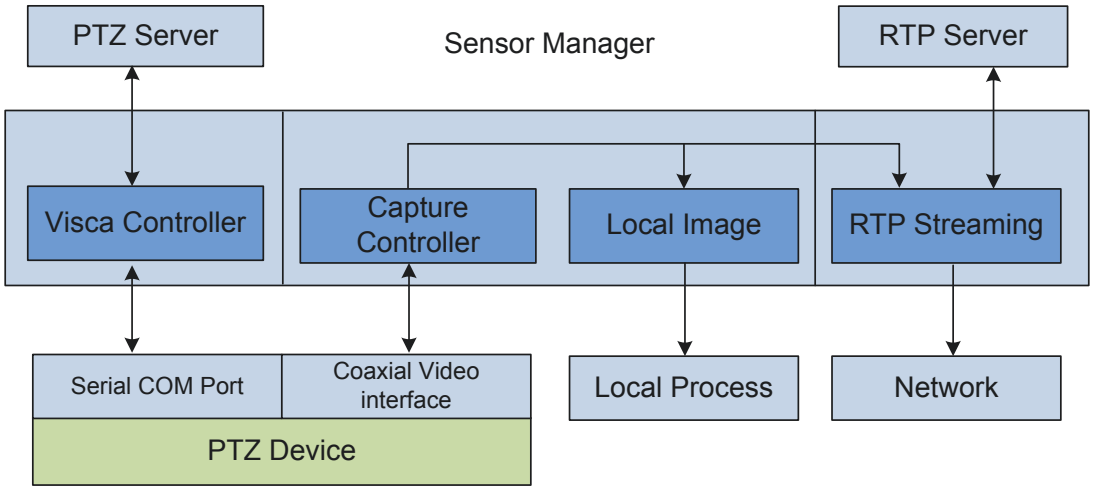


Figura 4.3: Arquitectura interna del controlador de sensor que hace de capa de abstracción hardware con las cámaras PTZ. Se puede observar las interfaces de transmisión de vídeo a través del protocolo RTP, así como del servidor PTZ que permite realizar movimientos en la cámara.

será capaz de realizar el procesamiento necesario para generar un conjunto de pistas presentes en el entorno.

Al estar tratando con un entorno real, ha sido necesario desarrollar el Controlador del Sensor para que las cámaras reales puedan ser integradas en el sistema multi-agente que estamos diseñando. Este controlador usa las tarjetas digitalizadoras Matrox Morphis instaladas en los servidores para digitalizar la imagen. Utiliza estas imágenes capturadas para detectar las entidades de color del entorno y hacer su seguimiento. También utiliza la compresión hardware que ofrecen para comprimir las imágenes antes de transmitirlas a través de la red. Además abre un puerto de comunicación serie que gestiona los comandos a través del protocolo VISCA

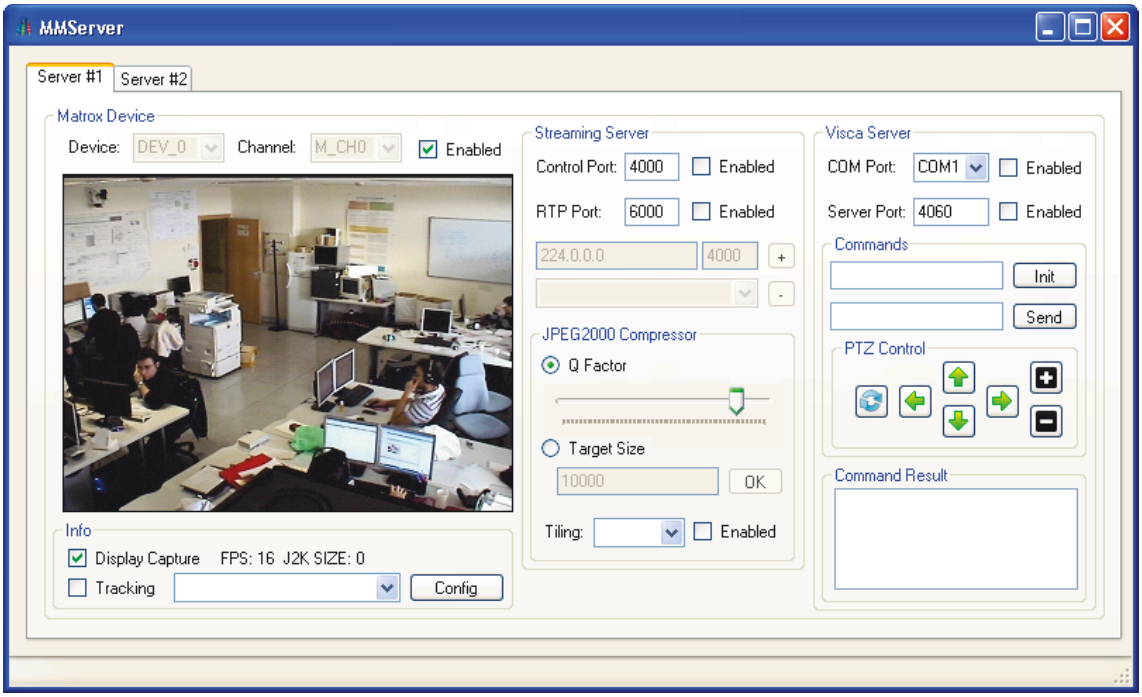


Figura 4.4: Interfaz de usuario principal del Controlador del Sensor. Permite configurar múltiples aspectos relacionados con la conexión de la cámara, la calidad de compresión de las imágenes, los algoritmos de seguimiento, etc.

para controlar el movimiento de la cámara. Todo esto se encapsula en un servidor TCP/IP que recibirá los comandos del *SA* que activará y desactivará las opciones mencionadas según vaya siendo requerido por otros agentes. Se puede observar el esquema interno de funcionamiento de este módulo en la figura 4.3.

A modo ilustrativo se presentan algunas capturas de la interfaz de usuario proporcionada por el Controlador del Sensor. Estas interfaces permiten configurar el canal donde se encuentra conectada la cámara (las tarjetas Matrox Morphis permiten hasta 16 canales), el puerto serie donde se encuentra conectada la tarjeta, los factores de compresión de JPEG-2000 que se aplicarán al transmitir las imágenes por la red, la configuración de los algoritmos de segmentación y seguimiento disponibles. Además permite visualizar los clientes que se encuentra conectados recibiendo información de seguimiento, controlando la cámara, y los destinos hacia donde se están transmitiendo las imágenes. Esta interfaz puede observarse en la figure 4.4.

El Controlador del Sensor ha sido diseñado para poder implementar diferentes algoritmos de seguimiento de manera transparente. En este momento se ha integrado un algoritmo de seguimiento de objetos por color, necesario para cumplir los objetivos propuestos. Este algoritmo es configurable desde la interfaz, permitiendo definir los colores que interesa detectar, así como determinados parámetros internos empleados. Esta interfaz se muestra en la figura 4.5. En ella se puede observar la imagen obtenida de la cámara, los elementos de configuración del

algoritmo, así como los elementos de color detectados en la escena.

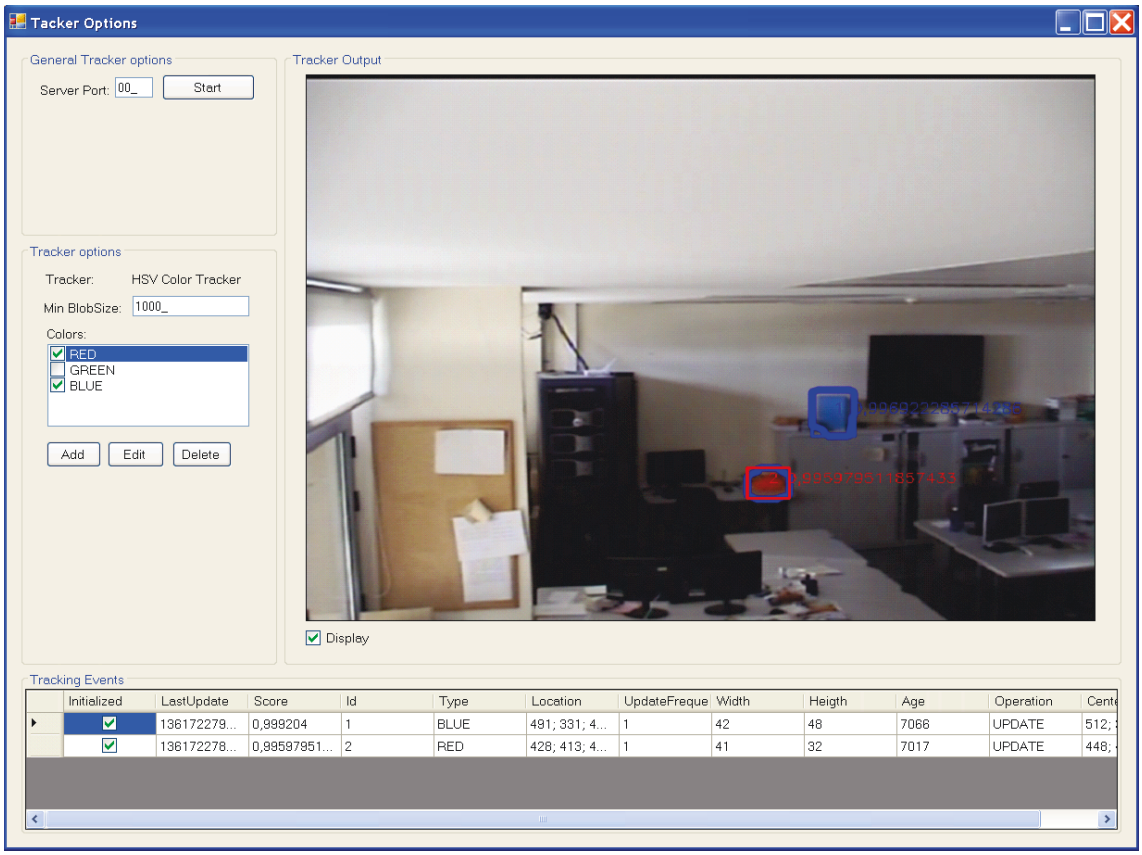


Figura 4.5: Interfaz de usuario del Controlador del Sensor referida a la configuración de un algoritmo de seguimiento. El que se muestra en concreto se trata de un algoritmo de seguimiento por color.

En cierto sentido, el Controlador del Sensor podría haber sido otro agente más de la arquitectura. Un agente especializado en el tipo de cámara que estuviera gestionando y que proporciona del mismo modo una interfaz para el resto de agentes. Desde el punto de vista teórico tendría mucho sentido, pero desde el punto de vista práctico, y dado que estamos tratando un problema real, no lo tendría tanto. Por ejemplo, tal y como está diseñado este componente software puede ser de utilizado para más proyectos, independientemente de que estén basados en arquitecturas multi-agente. Además, este componente requiere de librerías específicas para el acceso al hardware, y dependiendo del entorno multi-agente sobre el que vayamos a desarrollar la plataforma, podría ser complicada su integración. De esta forma nos estamos asegurando el correcto funcionamiento del componente, así como posibilitando su integración en otros proyectos.

#### 4.3.1.2 Integración Agente Sensor - Controlador del Sensor

Hemos querido desacoplar el uso del Controlador de Sensor del propio *SA* utilizando una conexión TCP/IP para su comunicación. De esta forma podemos desplegar el *SA* en cualquier parte, así resulta mucho más cómodo su desarrollo, distribución, y depuración. La interacción del *SA* con este controlador queda reflejada en la figura 4.6, donde se puede apreciar cómo el *SA* interacciona con las interfaces del Controlador de Sensor a través de un canal de comunicación TCP/IP. El resto de agentes que deseen acceder a la información generada por el sensor, deberá hacerlo por tanto a través del *SA* que lo representa.

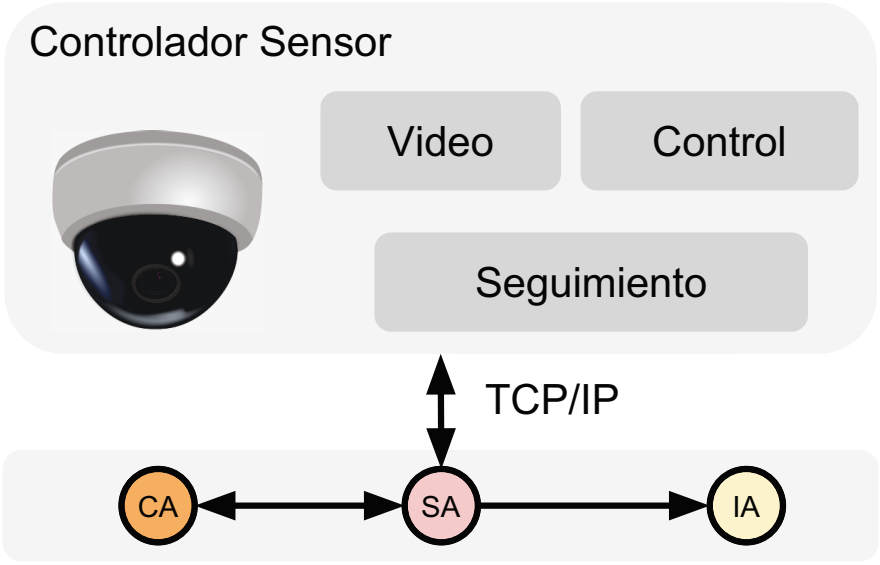


Figura 4.6: Interacción del *SA* con el Controlador del Sensor a través de una interfaz de comunicación TCP/IP. El Controlador del Sensor proporciona tres interfaces sobre la cámara que gestiona: una que transmite el vídeo de la cámara en tiempo real, otra que permite el control del movimiento, y otra que es capaz de proporcionar información sobre las entidades detectadas en el entorno.

Es importante remarcar que es el Controlador del Sensor el que realiza toda la interacción con la cámara, incluyendo el procesado de las imágenes para aplicar algoritmos de seguimiento. De esta forma es posible evitar flujos de vídeo innecesarios, que son costosos en términos de ancho de banda y procesado. Por ejemplo, un agente de tipo *SA* podría recibir la información de seguimiento proporcionada por el canal de comunicación TCP/IP y retransmitirla al resto de agentes que lo necesiten, como puede ser el *CA* o el *IA*. Estos agentes pueden utilizar esta información sin la necesidad de haber procesado un sólo fotograma de vídeo. Esto nos permite poder hacer llegar esta información a muchos más agentes, y que estos no tengan que realizar el procesado de las imágenes de forma independiente.

Por su parte, el flujo de vídeo generado por el Controlador del Sensor no pasaría directamente por el *SA*. El *SA* sería la interfaz a través de la cual el resto de agentes pueden solicitar la

transmisión del vídeo. El *SA* se encargaría de inicializar la transmisión hasta el agente que lo solicite. El flujo de vídeo por tanto iría directamente del Controlador del Sensor hasta el agente destino. Sobre la interfaz de control de la cámara, todas las peticiones serán gestionadas de igual forma a través del *SA*. Cualquier agente vinculado con el *SA* podría solicitar el movimiento de la cámara, por ejemplo, en función de la información de seguimiento recibida.

Este Controlador del Sensor, junto con el *SA* nos proporciona una gran flexibilidad a la vez que optimiza los recursos de ancho de banda y procesado necesarios para llevar a cabo determinadas tareas, como puede ser el seguimiento automático de objetos planteados en el objetivo.

#### 4.3.1.3 Descripción del Agente Sensor

El *SA* específico para el entorno multi-cámara propuesto, vista la descripción del Controlador del Sensor, así como su integración, resulta ser un agente que no requiere de una gran lógica de gestión interna. Este se limitará a ofrecer una interfaz para que el resto de agentes puedan interactuar con la cámara. Es decir, es como un elemento de software intermedio que permite la comunicación entre un entorno multi-agente y un servicio que se encuentra desplegado en un servidor.

Este agente permitirá abstraer al resto de agentes de la arquitectura de la implementación concreta que es necesaria para interactuar con la cámara. De esta forma es posible integrar cada cámara PTZ dentro de un entorno multi-agente distribuido, y que esta pueda transmitir las secuencias de vídeo a diferentes partes de la arquitectura, así como permitir la gestión de control que necesitamos en este tipo de entornos.

Por lo tanto podemos esperar de este agente un conjunto de capacidades que permiten trasladar las peticiones de los agentes del entorno, hacia la cámara al que se encuentra gestionando. La información de entrada y salida prevista para este agente se resumen en la tabla 4.2.

| | |
|---|---|
| **Entrada** | |
| **Petición de vídeo** | Generadas por cualquier agente presente en la arquitectura que requiera iniciar o detener un flujo de vídeo de la cámara. El flujo de vídeo se transmitirá por un canal de comunicación externo al utilizado por los agentes para comunicarse. |
| **Petición de control** | Generadas por cualquier agente de la arquitectura que necesite realizar un control sobre el campo de visión de la cámara. |
| **Petición de seguimiento** | Generadas por cualquier agente de la arquitectura que necesite información de seguimiento de los elementos de interés del escenario. |
| **Información de seguimiento** | Generada por el Controlador del Sensor cuando este se encuentre generando la información. Esta información sólo será recibida si el *SA* ha tramitado alguna petición de seguimiento de alguno de los agentes. Esta información se encuentra descrita en la tabla 4.1. |
| **Salida** | |
| **Petición de vídeo** | Transmitida desde los agentes hacia el Controlador del Sensor. |
| **Petición de control** | Transmitida desde los agentes hacia el Controlador del Sensor. |
| **Petición de seguimiento** | Transmitida desde los agentes hacia el Controlador del Sensor. |
| **Información de seguimiento** | Recibida del Controlador del Sensor y retransmitida a los agentes que la hayan solicitado. Esta información se encuentra descrita en la tabla 4.1. |

Tabla 4.2: Información de entrada y salida del Agente Sensor (SA) para el entorno multi-cámara.

### 4.3.2 Agente Fusión

El Agente Fusión (*FA*) aparece en la presentación de la arquitectura multi-agente de la figura 4.2 como un elemento intermedio entre los *SA* y los *CA*. Esto indica que este agente tendrá que tener la capacidad de integrar la información proporcionada por los *SA*. Como se puede observar en la figura 4.1, las cámaras se encuentra en un espacio demasiado pequeño, y es muy posible que las información independiente proporcionada por cada una de ellas esté representando a las mismas entidades. Por ello, si queremos realizar algún tipo de coordinación sobre los elementos

presentes en la escena necesitaremos incorporar un *FA* que genere un conjunto de entidades no redundantes sobre los que poder tomar decisiones.

Para diseñar el *FA* es necesario tener en cuenta la información que vamos a tratar de fusionar. En el escenario objetivo que hemos definido contamos con información de seguimiento generado por un algoritmo que detecta elementos de color. Para simplificar el problema, vamos a suponer que sólo vamos a encontrar un elemento de un determinado color en cada momento. No realizar esta simplificación nos supondría tener que tener otra variable para discernir las entidades, como pudiera ser su posición y velocidad. Es cierto que hay métodos de calibración que permiten obtener la posición en el mundo real de un objeto detectado en el plano de la cámara. Pero la mayoría de estos métodos están diseñados para cámaras estáticas. Las cámaras PTZ tienen la particularidad de que pueden cambiar el campo de visión, e inclusive el nivel de zoom, lo que cambiaría los parámetros de la óptica. Por este motivo es muy difícil encontrar un método de calibración genérico que podamos aplicar a nuestro entorno, ya que además de calibrar los parámetros ópticos, sería necesario calibrar otros parámetros como los ángulos de rotación de la cámara, su resolución, etc.

De este modo contaremos únicamente con la información de color proporcionada por los *SA*. Quizás en este *FA*, la fusión de datos como se la conoce tradicionalmente pase totalmente desapercibida, ya que no se está integrando información con la finalidad de mejorar las diferentes fuentes complementarias, ni se están aplicando algoritmos de estimación y filtrado ya que tampoco contamos con información de posición de las entidades. Sin embargo, el objetivo de este diseño es evaluar la arquitectura multi-agente para la coordinación en un entorno multi-cámara. Para este objetivo nos basta con entidades sencillas de detectar y de discernir. De todas formas, en el entorno marítimo descrito en el siguiente capítulo, sí se desarrolla un sistema más complejo y realista que incluye información de diferentes sensores.

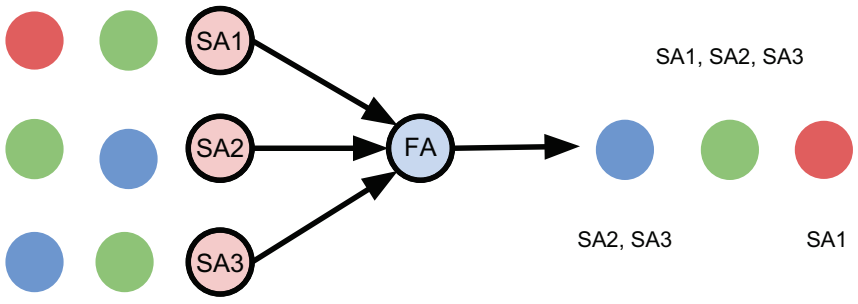


Figura 4.7: Agente Fusión (*FA*) empleando información de color de las pistas generadas por tres Agentes Sensor (*SA*) para producir una salida no redundante. Esta información de salida será de utilidad para tomar decisiones acerca de las pistas a monitorizar.

Para hacernos una idea de cómo se comportaría este agente de fusión podemos fijarnos en la figura 4.7. En esta contamos con los tres *SA* que representan las tres cámaras del entorno.

Estas cámaras estarían enviando información de seguimiento de las entidades de color. Como puede observarse, algunos sensores estarían en disposición de ver las mismas entidades (con el mismo color). El *FA*, conociendo la casuística del escenario será capaz de generar una salida fusionada, indicando las entidades presentes así como los sensores que contribuyen a cada una de ellas.

| | |
|---|---|
| **Entrada** | |
| **Información de seguimiento** | Información proporcionada por cada sensor de la información de seguimiento de la que disponen. Esta información incluirá como mínimo los siguientes elementos:<br>• Identificador del agente que ha generado el mensaje.<br>• Marca de tiempo del mensaje.<br>• Tipo de mensaje (evento de pista).<br>• Tipo de evento (creación, actualización, o borrado).<br>• Identificador local de la pista.<br>• Identificador de las fuentes de información que contribuyen a esta pista (compuesto por el identificador del agente y el identificador local de la pista).<br>• Atributos de la pista: En este entorno concreto contará el color y las dimensiones. Otros entornos podrían contar con diferentes atributos como posición, velocidad, forma, actividad, etc. |
| **Salida** | |
| **Datos Fusionados** | La información de salida dispondrá de los mismos elementos que la información de entrada descrita en esta tabla, pero actualizados por el *FA* con nuevos identificadores y un estado fusionado. De este modo, esta salida podría enviarse de forma transparente a otros agentes de fusión, como si la información estuviera siendo generada por otro sensor más. La pista fusionada incluirá además los identificadores de las pistas locales de cada agente que se han integrado. |

Tabla 4.3: Información de entrada y salida del Agente Fusión (FA) para el entorno multi-cámara.

La información de entrada y salida de este agente en el entorno multi-cámara se describe en la tabla 4.3. Básicamente este agente recibirá información de las pistas generadas por los diferentes sensores, y generará un conjunto de información similar pero fusionado. Esta

información se podrá utilizar directamente como si hubiese sido generada por un sensor más. Al fin y al cabo se sigue proporcionando información de seguimiento pero con los identificadores y el estado ligeramente alterados.

### 4.3.3 Agente Control

Un Agente de Control ($CA$) para un entorno multi-cámara como el propuesto podría implementar diferentes funcionalidades. Las más básicas podrían consistir en mover la cámara de forma aleatoria o por una ruta predefinida en busca de pistas en el entorno. Otra podría permitir al operador seleccionar una de las pistas presentes para realizar un seguimiento automático. También sería posible maximizar el número de pistas visualizadas en cada momento. Seguir pistas con determinadas características o que estén entrando en determinadas áreas del entorno. Todo esto dependerá del objetivo que nos propongamos resolver.

Nuestro objetivo concreto trata de un sistema de gestión autónomo que sea capaz de seguir las diferentes pistas ($P$) de un determinado color que se presenten en el entorno, intentando que cada color esté seguido por una cámara ($C$) en cada momento. Además, los colores tendrán determinadas prioridades, según se define en 4.1.

$$N_C \geq N_P \implies \forall P \exists C | P_{ROJO} > P_{AZUL} > P_{VERDE} \tag{4.1}$$

Hemos visto cómo los $SA$, a través del Controlador del Sensor son capaces de generar por si solos un conjunto de pistas actualizadas sobre las entidades detectadas en el entorno. Así mismo, el $FA$ es capaz de recoger toda esta información e integrarla con el fin de proporcionar un conjunto de pistas únicas. De este modo, el $CA$ será el responsable de controlar los movimientos de la cámara de acuerdo a los objetivos establecidos y las percepciones recibidas del entorno a través del $FA$.

Este control se refleja en la figura 4.8, donde un $CA$ utiliza la información recibida por el $FA$ para controlar la cámara y coordinarse con el resto de agentes a la hora de cumplir el objetivo. Los beneficios de utilizar un $SA$ que es capaz de generar información refinada sobre el entorno, es que este agente no tiene que lidiar con la gestión ni el procesamiento de la cámara a bajo nivel. Únicamente utiliza la información recibida por el sensor, pasando antes por el sistema de fusión, para realizar el control en tiempo real.

La coordinación con el resto de agentes $CA$ se realiza mediante un sistema basado en eventos. Cuando un CA comienza a monitorizar una determinada pista $P$, este envía un evento al resto de agentes para notificar la acción. La pista $P$ siempre irá referida a las pistas generadas por el sistema de fusión, de este modo, el resto de agentes que puedan estar observando la misma pista, sepan de qué entidad se trata. Esto les permitirá coordinarse sobre las diferentes pistas que aparecen en el entorno.

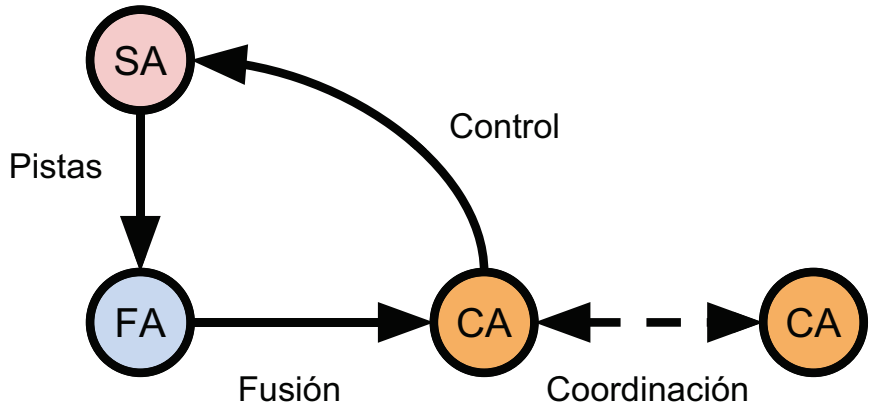


Figura 4.8: Agente Control (*CA*) para el entorno multi-cámara. Recibirá información fusionada a través de un Agente Fusión (*FA*) que utilizará para controlar el Agente Sensor (*SA*) asociado. Este agente también necesitará coordinarse con otros agentes para llevar a cabo el objetivo global.

Cuando un *CA* detecta una nueva pista *P* del entorno que tiene más prioridad que la que ya estuviera monitorizando, entonces notifica al resto de *CA* su intención tentativa de monitorizar la nueva pista *P*. Para poder lanzar este evento tentativo, no solo la pista *P* tiene que tener más prioridad, si no que además, la acción del agente no suponga incumplir con el objetivo planeado. Para ello cada *CA* necesita conocer el estado actual del resto de *CA* (conocido a través de los eventos generados) para evaluar si su acción no supone un incumplimiento de los objetivos. Pasado un período de espera breve, si no se ha recibido ningún otro evento, ya sea tentativo o definitivo, que plantee revaluar la acción propuesta, esta se confirma. La confirmación lleva asociado también otro evento de confirmación.

Se podría dar el caso de una pista que aparece a la vez para todos los *CA*, y a su vez tiene más prioridad que las pistas que ya estuvieran monitorizando. Todos los *CA* evaluarán en este momento si pueden hacer el cambio a la nueva pista, y su resultado será positivo. En ese momento los *CA* emitirán un evento de monitorización tentativo indicando su intención. En el período de espera de cada acción tentativa, cada agente habrá recibido la intención del resto de agentes. Será en este instante donde cada agente revalúa la acción tentativa propuesta. En esta ocasión, cada agente por separado se dará cuenta que si todas las acciones tentativas se aplican, alguna pista de las existentes se quedaría sin monitorizar, lo que supondría incumplir el objetivo.

En este instante los agentes se podrían coordinar de diferentes modos para resolver quiénes confirmarán la acción tentativa. Un enfoque distribuido podría consistir en la formación de una coalición dinámica entre los agentes involucrados. En ella podrían tomar una decisión en función de diferentes parámetros, como el tamaño de la pista que observa cada uno, quién tomó antes la iniciativa, el ángulo libre que tiene para seguir monitorizándola, etc. Un enfoque centralizado consistiría en el uso de un agente mediador que evaluaría las propuestas de cada

uno de los agentes y ofrecería una solución. El enfoque que seguiremos en este diseño, por simplificar el desarrollo, consistirá en que los agentes trabajen sobre una serie de reglas de antemano para solucionar este tipo de conflictos. En concreto consiste en que cada agente que detecte un estado tentativo que no cumpla con el objetivo o lo empeore, evaluará las acciones tentativas del resto de los agentes. Si la suya es posterior a cualquiera de ellas, se descarta automáticamente. Si es la primera en haberse lanzado, se asume que el resto descartarán sus acciones tentativas. En el hipotético caso de que además las acciones tentativas generadas contengan el mismo instante de tiempo (a nivel de milisegundo), estás se resolverían por el nombre del agente, que es único en el despliegue de la arquitectura.

Para que este mecanismo de coordinación funcione correctamente, todas las máquinas donde se están ejecutando los agentes deberían encontrarse sincronizadas en el tiempo. Para ello las máquinas contarán con protocolos de sincronización de tiempo como NTP (Network Time Protocol). De esta forma los agentes contarían con la misma base de tiempos (aproximadamente) para comparar los tiempos de las acciones tentativas.

A modo ilustrativo del caso peor donde todas las cámaras comienzan a ver las mismas pistas en el mismo instante de tiempo, se muestra la tabla 4.4. En esta se puede observar cómo sería el estado interno de los agentes (todos comparten la misma información gracias a los eventos que se transmiten). Cada CA almacena una matriz que tiene por filas las pistas detectadas del entorno, y por columnas a todos los CA. Cada agente, en función del resto de eventos recibidos por el resto de agentes, va marcando la acción que está realizando cada agente. En la tabla, las $X$ representarían que un CA está siguiendo a una pista determinada. Un $*$ representaría un evento tentativo de seguimiento, o lo que es lo mismo, el CA comenzará a seguir la pista si no encuentra ningún conflicto tras un determinado período de tiempo.

En este ejemplo se mostrarían el estado de los tres CA ($CA_1$, $CA_2$, y $CA_3$) en diferentes instantes de tiempo según van apareciendo nuevas pistas. Las pistas que aparecen siguen una prioridad, siendo $P_0 > P_1 > P_2$. Estas representarían el caso real propuesto con las prioridades de los colores. En el instante $T_0$ no hay ninguna pista, por lo que ningún agente tiene ninguna asignación. En el instante $T_1$ aparece la pista $P_1$. Aunque estas pistas sean el resultado de la fusión, y por tanto únicas, los agentes pueden saber si son capaces o no de seguirlas teniendo en cuenta que las pistas fusionadas contienen también la información local de cada una de las aportaciones. En $T_1$ sólo está disponible la pista $P_1$ que todos los $CA$ ven, entonces comienzan a seguirla. En $T_2$ aparece la pista $P_0$ que tiene más prioridad que $P_1$. Como estamos en el caso peor, y siguiendo el algoritmo descrito en 1, todos los $CA$ de forma simultánea lanzarán sus acciones tentativas de seguimiento. En el período tentativo de cada uno de los agentes estos descubrirían el conflicto. De confirmarse el nuevo estado tentativo, la pista $P_1$ dejaría de estar cubierta. En este instante, los agentes $CA_2$ y $CA_3$ con menos prioridad (por el nombre), deciden cancelar su acción tentativa. El agente $CA_1$ la confirma sin embargo. En el siguiente instante $T_3$, los agentes $CA_2$ y $CA_3$, al volver a evaluar las pistas disponibles, ambos deciden cambiar

| Instante | Pista | $CA_1$ | $CA_2$ | $CA_3$ | Eventos |
|---|---|---|---|---|---|
| $T_0$ | - | - | - | - | |
| $T_1$ | $P_1$ | X | X | X | $CA_1$ - $P_1$ Confirma<br>$CA_2$ - $P_1$ Confirma<br>$CA_3$ - $P_1$ Confirma |
| $T_2$ | $P_0$ | * | * | * | $CA_1$ - $P_0$ Tentativo<br>$CA_2$ - $P_0$ Tentativo<br>$CA_3$ - $P_0$ Tentativo |
| | $P_1$ | X | X | X | |
| $T_3$ | $P_0$ | X | | | $CA_1$ - $P_0$ Confirma<br>$CA_2$ - $P_0$ Cancela<br>$CA_3$ - $P_0$ Cancela |
| | $P_1$ | | X | X | |
| $T_4$ | $P_0$ | X | * | * | $CA_2$ - $P_0$ Tentativo<br>$CA_3$ - $P_0$ Tentativo |
| | $P_1$ | | X | X | |
| $T_5$ | $P_0$ | X | X | | $CA_2$ - $P_0$ Confirma<br>$CA_3$ - $P_0$ Cancela |
| | $P_1$ | | | X | |
| $T_6$ | $P_0$ | X | X | | |
| | $P_1$ | | | X | |
| | $P_2$ | * | * | | $CA_1$ - $P_2$ Tentativo<br>$CA_2$ - $P_2$ Tentativo |
| $T_7$ | $P_0$ | | X | | |
| | $P_1$ | | | X | |
| | $P_2$ | X | | | $CA_1$ - $P_2$ Confirma<br>$CA_2$ - $P_2$ Cancela |

Tabla 4.4: Representación de estado de los Agentes Control (CA) cuando aparecen pistas en el entorno en el caso peor. Las prioridades de las pistas son $P_0 > P_1 > P_2$. Los eventos se estarían generando en el mismo instante y se estarían resolviendo por el nombre del agente, con prioridad $CA_1 > CA_2 > CA_3$. Las X representan que un CA está siguiendo a una pista determinada. Los * que el agente propone tentativamente seguir a la pista marcada.

a la pista $P_0$ porque es más prioritaria. Esto no supone un conflicto cuando internamente lo evalúan, porque cada uno sabe que hay otro agente que está monitorizando la pista $P_1$. Por lo tanto en $T_4$ se repetiría el mismo proceso que en el caso anterior. El único que confirmaría sería $CA_2$, mientras que $CA_3$ cancela su acción en $T_5$. Para finalizar, en $T_6$ aparece una nueva pista. En este caso es la menos prioritaria que las demás. Sin embargo, la lógica de los agentes detectan que aunque se encuentren monitorizando una pista más prioritaria, hay otra que no está siendo cubierta. Los dos únicos agentes que pueden llevar a cabo esta acción serían los agentes $CA_1$ y $CA_2$, que ambos saben que la pista que ellos siguen está al menos cubierta por otro agente. Por ello ambos notifican su acción tentativa. Igual que en el resto de pasos, la única en confirmarse es la de $CA_1$ en $T_7$.

**Algoritmo 1** Algoritmo para la selección de objetivos de monitorización en función de su prioridad y el estado del resto de agentes.

**Entrada:** Actualización de las pistas del entorno
**Entrada:** Estado de monitorización de los agentes
**Salida:** Selección del objetivo a monitorizar

```
si hay alguna pista sin monitorizar entonces
    si tiene mas prioridad que la que estoy monitorizando actualmente entonces
        devolver pista sin monitorizar
    si no
        si hay otro agente cubriendo la pista que yo estoy monitorizando entonces
            devolver pista sin monitorizar
        fin si
    fin si
si no
    si puedo monitorizar otra pista más prioritaria que la mía entonces
        si mi pista está siendo por otro agente entonces
            devolver pista más prioritaria
        fin si
    fin si
fin si
devolver pista actual
```

Hay que recalcar que esto sólo ocurriría en el caso peor. En la realidad los más probable es que un agente notifique la acción tentativa y esta llegue al resto de agentes. Esto supone que el resto de agentes perderían capacidad de iniciativa, ya que otro ha sido el que antes ha visto la pista, y por ello no puedan tomar la misma decisión. Por lo que el número de eventos transmitidos y los conflictos a resolver serían mucho menores.

Visto el mecanismo de coordinación basado en eventos disparados por la información generada por la fusión, y la coordinación entre los agentes basada en un mecanismo basado en reglas, podemos pasar a describir la información de entrada y salida de cada agente *CA*, tal y como se muestra en la tabla 4.5.

| | |
|---|---|
| **Entrada** | |
| **Información de coordinación** | Eventos de monitorización generador por otros agentes *CA* que indican la intención de monitorización de determinados objetivos producto de la de fusión. |
| **Pistas Locales Visión** | Conjunto de pistas locales generadas por la cámaras PTZ (a través del *SA* asociado). Estas podrían ser útiles en algún modo de control autónomo que no dependa de la información de fusión ni la coordinación con otros agentes. |
| **Pistas Globales** | Conjunto de pistas globales generadas por el *FA* tras el proceso de integración de una o múltiples pistas locales proporcionadas por cada *SA*. Esta información se utilizará principalmente para la selección de objetivos y coordinación con el resto de agentes. |
| **Salida** | |
| **Información de Control** | Control realizado sobre la cámara PTZ que se encuentra gestionando. Este control se generará en función de las detecciones percibida de la fusión, y el mecanismo de priorización y coordinación con los que cuenta el agente. |
| **Información de Coordinación** | Eventos de monitorización generados por el agente para notificar al resto de agentes homólogos los eventos de seguimiento. |

Tabla 4.5: Información de entrada y salida del Agente Control (*CA*) para el entorno multi-cámara.

### 4.3.4 Agente Operador

Para este escenario, donde a priori sólo contamos con un objetivo a realizar, no será necesario plantear el diseño de un Agente Operador (*OA*) que gestione los diferentes modos de funcionamiento del sistema de vigilancia a través de los *CA*. Tendría sentido diseñarlo si contamos con diferentes modos que pueden ser alternados o priorizados en función de los objetivos establecidos por el operador. O también si necesitamos interactuar con la plataforma multi-agente a través de algún sistema remoto que inicie o pare las tareas de vigilancia autónomas. En este caso, como sólo hay un tipo de *CA* diseñado para realizar el objetivo de monitorización, estos estarán funcionando por defecto en este modo. Por lo que no será necesario desplegar un *OA* para controlarlos.

### 4.3.5 Agente Interfaz

La interfaz de usuario es probablemente una de las partes más importantes de un sistema de videovigilancia. Esta es la parte de interacción con el operador que se encuentra monitorizando y tomando decisiones sobre las acciones que ocurren en el entorno. Estos sistemas están compuestos normalmente por un conjunto de pantallas que permiten monitorizar en tiempo real la información generada por los diferentes sensores de visión.

En una aplicación como la que estamos diseñando en esta sección, además podríamos considerar incorporar información adicional, como la proporcionada por el sistema de fusión. Esta permitiría mostrar en la interfaz determinada información sobre el número de objetivos detectados, su ubicación en cada una de las cámaras, el detalle de cada objetivo, etc. Un ejemplo de esta interfaz se podría observar en la figura 4.9. Con ella, el operador tendría a su disposición más información para poder tomar las decisiones oportunas.

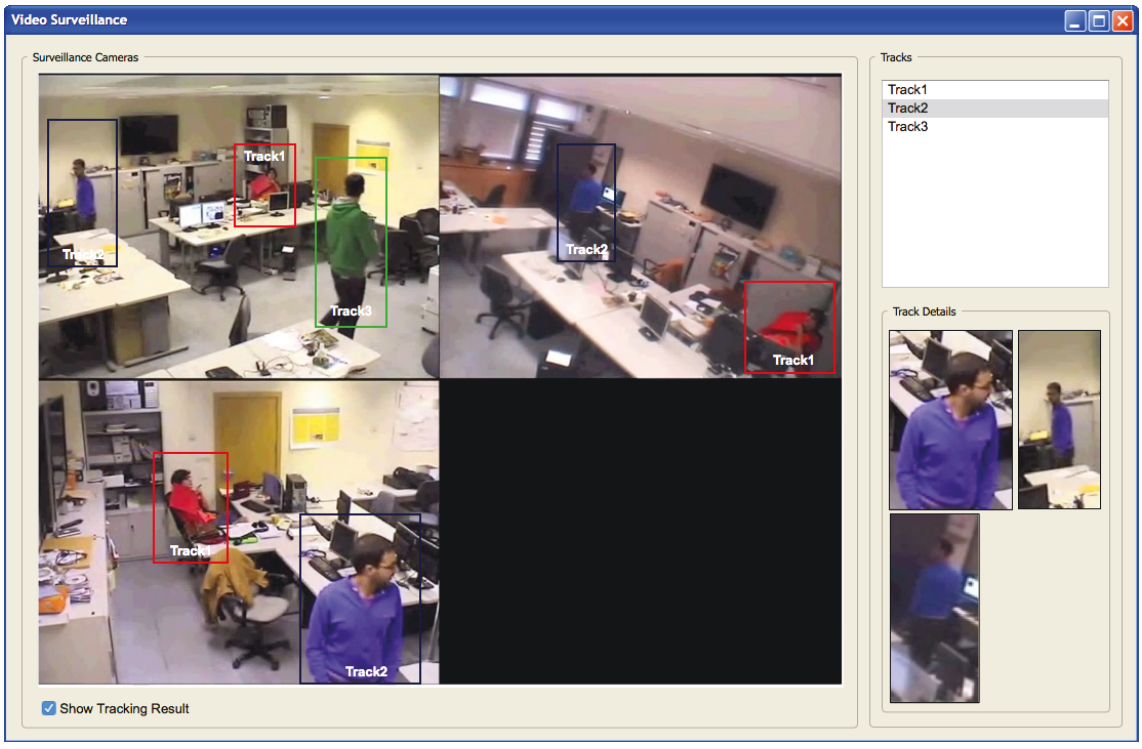


Figura 4.9: Ejemplo de representación de una interfaz de usuario para el entorno de vigilancia multi-cámara. Esta permitiría representar las diferentes fuentes de vídeo obtenidas del entorno, así como información adicional percibida del sistema de fusión, que permitirá identificar cada uno de los objetivos y centrar la atención en cada uno de ellos.

Se puede observar en esta figura la representación de tres cámaras monitorizando diferentes parte del entorno, y sobre esta señal de vídeo se muestra la superposición de la información generada por los algoritmos de seguimiento y la fusión, como el identificador de cada pista así

como su posición dentro de la imagen. El operador además podría llegar a seleccionar una pista en particular para visualizar sus detalles, como el estado de la pista, el tiempo de vida, etc.

Esta tarea será llevada a cabo por el Agente Interfaz (*IA*), que será el encargado de recibir diferentes flujos de información de la arquitectura multi-agente para su visualización por parte del operador. Podemos identificar diferentes entradas en el *IA*, como la información de vídeo generada por cada una de las cámaras, así como el resultado de la fusión de información que integra las pistas locales generadas por cada cámara. Este esquema quedaría representado por la figura 4.10.

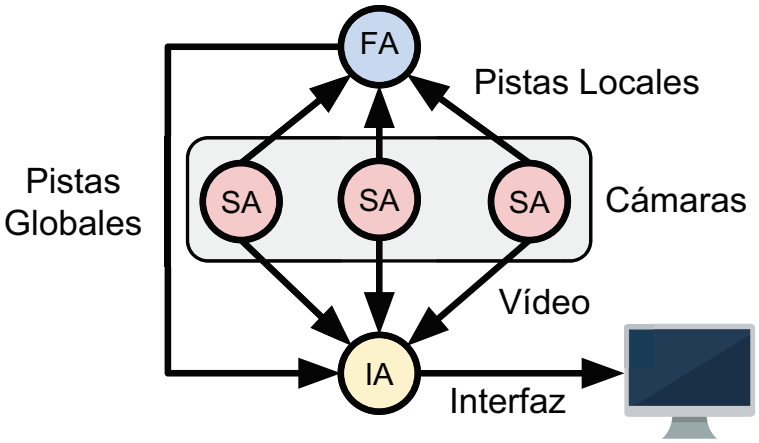


Figura 4.10: Representación del Agente Interfaz (*IA*) para el entorno multi-cámara. Recibirá los flujos de vídeo y la información de fusión para su visualización por parte del operador.

La información de entrada y salida del agente *IA* quedaría reflejada en la tabla 4.6. Esta podría variar si no fuera necesario o útil disponer de la información de fusión en la representación del entorno.

| | |
|---|---|
| **Entrada** | |
| **Vídeo** | Flujos de vídeo proporcionados por los diferentes *SA* que permiten al operador visualizar el entorno. |
| **Resultado de la Fusión** | Sería el conjunto de pistas globales proporcionado por el sistema de fusión para que el operador puede tener una imagen global de la situación del entorno.. |
| **Salida** | |
| **Interfaz Gráfica** | Podría tratarse de una interfaz gráfica como la mostrada en la figura 4.9, que visualice los diferentes flujos de vídeo así como determinada información de Fusión. |

Tabla 4.6: Información de entrada y salida del Agente Interfaz (*IA*) para el entorno multi-cámara.

## 4.4 Optimización de la integración de sensores de vídeo

Durante el desarrollo experimental de la aplicación multi-cámara descrita en esta sección, nos encontramos con un conjunto de cámaras PTZ convencionales que estaban conectadas a una serie de servidores. Cada uno de estos servidores cuenta con hardware digitalizador de vídeo que permite convertir la señal analógica de la cámara, a una señal digital para ser procesada por los ordenadores. Estas tarjetas digitalizadoras nos permitían el proceso de captura de la imagen de las cámaras, y por tanto el uso de las imágenes para los procesos de aplicación de algoritmos de seguimiento, transmisión, y visualización, como se ha descrito en la sección 4.3.1.1.

Estas tarjetas digitalizadoras, además, permiten la compresión por hardware de las imágenes a través del algoritmo JPEG2000. Por ello se contempló el uso de esta característica para integrar la parte de transmisión de vídeo a los Agentes Interfaz de la arquitectura. Esto nos permitiría soportar la transmisión de secuencias en tiempo real, necesario para un entorno de videovigilancia. Se podría haber pensado en utilizar otras técnicas de compresión de vídeo, pero teniendo hardware especializado para esta tarea, puede resultar muy provechoso su utilización.

El formato JPEG2000 soportado por las tarjetas, que a pesar de sus beneficios es muy poco utilizado hoy día, nos dio lugar a pensar en diferentes técnicas de optimización para realizar una transmisión de vídeo todavía más eficiente. Este desarrollo se convirtió en una tarea adicional del proceso de integración del proceso multi-cámara, y permitió la publicación de diversos artículos en congresos y revistas, e inclusive una patente.

Por este motivo, pasaremos a describir el proceso de optimización que realizamos con las cámaras PTZ del entorno multi-cámara para sacar partido del hardware disponible.

### 4.4.1 Introducción

La transmisión de vídeo sobre redes de comunicación IP se ha extendido de manera significativa en los últimos años. La creciente capacidad de estas redes para transportar grandes volúmenes de información, las han convertido en el medio idóneo para poder transmitir múltiples secuencias de vídeo de forma simultánea. En este ámbito han surgido numerosas tecnologías y códecs de compresión de vídeo que permiten la transmisión de manera eficiente. Por ejemplo, el sistema Motion J2K (MJ$P_2$) (Boliek, 2000) es un sistema de compresión de vídeo (ISO/IEC, 2000; Skodras et al., 2001) basado en JPEG2000 (también conocido como J2K). Este emplea un sistema de compresión intra-frame (comprime cada fotograma por separado), por lo que se evita el uso de los algoritmos de compensación de movimiento que encontramos en múltiples estándar, como el MPEG-2 (Haskell, 1997), o el MPEG-4 (Ebrahimi and Pereira, 2002).

Los códces como el MPEG-2 y el MPEG-4 o incluso el Motion JPEG (MJPEG) son los códecs más comunes y que más han sido probados durante las últimas décadas. Sin embargo, el códec MJ$P_2$ se empezó a integrar en los sistemas de vigilancia sobre el año 2000 (Charrier et al., 1999; Fukuhara et al., 2000), tras la aparición pública del códec. Estos sistemas tienen la ventaja de las características que puede proporcionar el códec J2K, como mejor tolerancia a fallos, definición de regiones de interés, y soporte de decodificación en múltiples resoluciones y calidad (Christopoulos et al., 2000; Santa-Cruz and Ebrahimi, 2000).

El sistema MJ$P_2$ es además el estándar adoptado actualmente para almacenar, distribuir, y reproducir las películas de alta definición en los cines digitales (Initiative et al., 2008). Es un estándar ISO definido como mejora del tradicional sistema de compresión MJPEG, basado en el códec que le da su nombre: JPEG (Pennebaker and Mitchell, 1993).

Por lo tanto, esta solución tecnológica debería permitirnos aplicaciones de transmisión de vídeo de mayor calidad y resolución junto con menores anchos de banda (Fossel et al., 2003; Marpe et al., 2004; Shirai et al., 2006). Las áreas de aplicación podrían incluir entre otras: cine digital, vigilancia remota, sistemas médicos de alta resolución, imagen satelital, etc. En la transmisión de vídeo en tiempo real tenemos la ventaja de que podemos comprimir cada fotograma de manera independiente con el códec J2K (ISO/IEC, 2000), y transmitir la imagen directamente sin esperar a procesar más fotogramas, reduciendo los períodos de latencia del sistema. Sin embargo, tiene como desventaja que esta latencia tan baja se consigue a costa de incrementar las necesidades de ancho de banda. Esto es porque no se realiza ninguna técnica reducción de redundancia temporal, como los algoritmos de compensación de movimiento presentes en los sistemas de compresión tradicionales.

Actualmente podemos encontrar algunos trabajos que tratan de optimizar la transmisión de secuencias de vídeo basadas en el códec J2K. El primer trabajo describe una propuesta basada en el análisis de la escena para reducir el ancho de banda y la complejidad en la transmisión (Meessen et al., 2005). Esto lo realizan distinguiendo los elementos del fondo con los de primer plano. Estos elementos se comprimirían a diferentes resoluciones, por ejemplo, seleccionando una resolución mejor para los objetos más cercanos o más importantes de la imagen. Sin embargo no se detalla el proceso de segmentación para discernir los elementos del fondo con los del primer plano, ni cómo se aplica sobre la secuencia de vídeo J2K.

El trabajo descrito en (Devaux et al., 2007; Naman and Taubman, 2007a,b), es más similar a la optimización presentada en esta tesis. En este trabajo se busca utilizar técnicas de compensación de movimiento para evitar la transmisión de pequeñas partes de la imagen (llamadas code-blocks). Este enfoque parece óptimo, pero parece demasiado complejo para ser aplicado en sistemas de tiempo real. Además, tanto el servidor como el cliente de estas secuencias de vídeo tendrían que soportar específicamente esta técnica. Que por otra parte no puede ser integrada en un estándar de transmisión de vídeo J2K como el descrito en (Futemma

et al., 2008). Tampoco se discute el retardo introducido por la técnica de compensación de movimiento, ni si se ha evaluado en un entorno real.

En nuestro caso, proponemos un sistema de compresión de vídeo basado en J2K, que llamamos Motion Interframe J2K (MIJ2K), que introduce técnicas de compensación de movimiento que pueden ser aplicadas a secuencias de vídeo en tiempo real. Además, la técnica propuesta puede ser integrada, y de hecho se implementará, siguiendo el estándar de transmisión (Futemma et al., 2008). Este método nos permitirá mejorar sustancialmente el ancho de banda necesario para la transmisión de vídeo, mientras mantenemos la ventaja de la baja latencia proporcionada por estos sistemas. En el trabajo que realizamos en (Luis and Patricio, 2009), ya describimos la transmisión de vídeo en tiempo real usando J2K, implementado sobre el estándar RFC 5371 (Futemma et al., 2008). Pero aún no se describía ningún mecanismo de optimización de las secuencias de vídeo, como el que se plantea ahora.

Por lo tanto, en esta sección se describe la integración de un mecanismo de compensación de movimiento en sistemas de transmisión basados en J2K. Esto nos permitirá obtener una mejor calidad en la imagen para el mismo ancho de banda. Para evaluar el método presentado, se comparará con sistemas de compresión actuales como el MJ$P_2$, MJPEG, o H.264-Intra. Comprobaremos la viabilidad de adecuación de nuestro método a su aplicación en sistemas de tiempo real midiendo la latencia introducida. Para evaluar la mejora en calidad utilizaremos métricas como la relación señal/ruido (PSNR), la similitud estructural (SSIM) y la métrica de calidad visual (VQM).

En el resto de la sección se especifica el método desarrollado, así como las pruebas realizadas. Comenzaremos explicando algunas características del códec de compresión J2K, seguido de una descripción de las técnicas de evaluación de compresión de vídeo. Finalmente describimos la arquitectura y realizaremos los experimentos de comparación con otras alternativas actuales.

### 4.4.2 Estructura interna J2K

El sistema J2K es un estándar de compresión basado en transformadas de ondícula (del término en inglés wavelet transformation) (Adams and Ward, 2001). Este fue diseñado y desarrollado por el comité *Joint Photographic Experts Group* (JPEG) en el año 2000, con la idea de reemplazar al conocido formato JPEG basado en transformadas de coseno, que databa de 1992.

Aunque el J2K ofrece un modesto incremento en la capacidad de compresión cuando lo comparamos con el tradicional JPEG, su principal beneficio es la flexibilidad que ofrece el resultado de la compresión (code-stream). Este es escalable, en el sentido de que puede ser decodificado de múltiples formas. Por ejemplo, podemos decodificar la imagen a diferentes niveles de calidad o resolución decodificando más o menos información. De esta forma, las aplicaciones podrían seleccionar la calidad o resolución que necesitan. Por ejemplo, si contamos

con una imagen de 30 megapíxeles, y queremos abrirla en un dispositivo móvil, quizá no tenga sentido decodificarla a su máxima resolución. Con J2K podríamos decodificar la cantidad de información necesaria para obtener una representación de la imagen que se adecúe a nuestro dispositivo, reduciendo la capacidad de procesamiento necesaria.

Algunas de las características más importantes de estas imágenes las podemos describir a continuación:

- Mayor rendimiento de compresión: A baja compresión, cuando los artefactos generados por la compresión son prácticamente imperceptibles, J2K tiene una pequeña mejora medible sobre JPEG. Con alta compresión (por ejemplo, menos de 0.25 bits/pixel), J2K tiene una gran ventaja sobre JPEG: Los artefactos son mucho menos visibles y apenas se ven los bloques. Esto se atribuye al uso de DWT (Discrete Wavelet Transform) (Santa-Cruz et al., 2000).
- Múltiples resoluciones de decodificación: J2K descompone la imagen en una representación de múltiples resoluciones, cada con su propio proceso de compresión. Lo que permite decodificar las resoluciones que sean necesarias (ISO/IEC, 2000).
- Decodificación progresiva con escalabilidad en relación señal/ruido (SNR): Esto permite decodificar una representación completa de la imagen a baja calidad e ir incrementado su calidad según se va recibiendo más información. Esto sería muy útil por ejemplo en la visualización de imágenes en páginas web. El estándar de JPEG de 1991 también contaba con esta característica pero su utilización es bastante infrecuente.
- Compresión con y sin pérdida: Igual que sucede con JPEG (Wallace, 1991), el estándar J2K proporciona la capacidad de compresión con y sin pérdida en una única arquitectura de compresión. La compresión sin pérdida es proporcionada por el uso de una transformación reversible de ondícula.
- Acceso y procesamiento aleatorio: J2K proporciona múltiples mecanismos para soportar el acceso aleatorio a regiones de interés con diferentes niveles de granularidad. Esta característica está soportada en parte por el concepto de cuadrícula (tiling) introducido que incorpora el nuevo estándar. Con esta característica, se puede dividir una imagen en múltiples partes que pueden ser codificadas y decodificadas por separado. Por cada pequeña parte existen además otros mecanismos de acceso aleatorio.
- Tolerancia a fallos: Igual que JPEG, J2K es robusto frente a la alteración de los datos binarios, principalmente porque los datos se codifican en pequeños bloques independientes.
- Soporte para transparencias: A diferencia de JPEG, este formato sí soporta por completo múltiples canales de transparencia como el conocido formato de intercambio Portable Network Graphics (PNG).

Aparte de estas características, el mayor beneficio de utilizar J2K para transmitir secuencias de vídeo en tiempo real, es que a diferencia de otros compresores como MPEG-4, la compresión y transmisión se puede realizar en tiempo real, tal como demostramos en el trabajo presentado (Luis and Patricio, 2009). Lo que nos permite utilizar este estándar de compresión en entornos como el de la videovigilancia con el que estamos trabajando.

Para esta propuesta en particular, la gran ventaja que hemos considerado, es que una imagen puede ser opcionalmente partida en múltiples regiones rectangulares no solapadas, llamadas cuadrículas (o tiles en inglés) (ISO/IEC, 2000). En esta propuesta se explotará esta característica, que únicamente proporciona este estándar de compresión, para aplicar un algoritmo en tiempo real de compensación de movimiento. Para ello nos basaremos en una técnica de diferencia basada en bloques (Shi and Sun, 1999), pero adaptando el rol de los bloques a las cuadrículas.

Las cuadrículas pueden ser de cualquier tamaño, inclusive del tamaño total de la imagen. De hecho una imagen que no define un tamaño específico de cuadrícula contendrá una única cuadrícula con toda la imagen. Si se especifica un tamaño de cuadrícula menor al tamaño total de la imagen, ésta quedará dividida automáticamente en un conjunto de pequeñas cuadrículas del mismo tamaño. Excepto en los límites inferior y derecho de la imagen, donde pueden aparecer cuadrículas algo más pequeñas. Dividir una imagen en múltiples regiones tiene algunas ventajas, como que el proceso de codificación y decodificación se puede realizar consumiendo menos memoria. Además nos permite un control total sobre qué partes de la imagen vamos a comprimir, transmitir, descomprimir, etc.

Sin embargo, siempre existen ventajas y desventajas con el uso de diferentes tamaños de cuadrícula. Como se describe en (Varma and Bell, 2004), un tamaño de cuadrícula pequeño reducirá la eficiencia de algoritmo de compresión J2K, lo que limitaría el interés del uso de un algoritmo de DWT. Además puede crear artefactos cuadrados, similares a los de JPEG, cuando aplicamos una gran relación de compresión. En este trabajo utilizaremos un tamaño relativamente pequeño de cuadrícula, ya que la peor eficiencia en compresión es rápidamente compensada por la técnica de compensación de movimiento, como describimos en el trabajo presentado en (Luis Bustamante et al., 2009).

En la figura 4.11 presentamos un ejemplo de la estructura interna de una imagen comprimida con J2K y cómo queda dividida en diferentes regiones o cuadrículas. La primera marca que encontramos es una marca de comienzo de información, llamada Start of Code Stream (SOC). Después de esta marca nos encontraremos la cabecera principal (Main Header), que incluye información necesaria para decodificar la imagen. La siguiente marca, conocida como Start Of Tile delimita el comienzo de una nueva cuadrícula. Cada cuadrícula contendrá una cabecera (Tile Part Header), y una parte de datos (Tile Part Bitstream). Podemos encontrar tantas cuadrículas como sean necesarias para cubrir la imagen completa. Y estas siempre serán del mismo tamaño (menos en los bordes de la imagen dependiendo del tamaño de la cuadrícula y

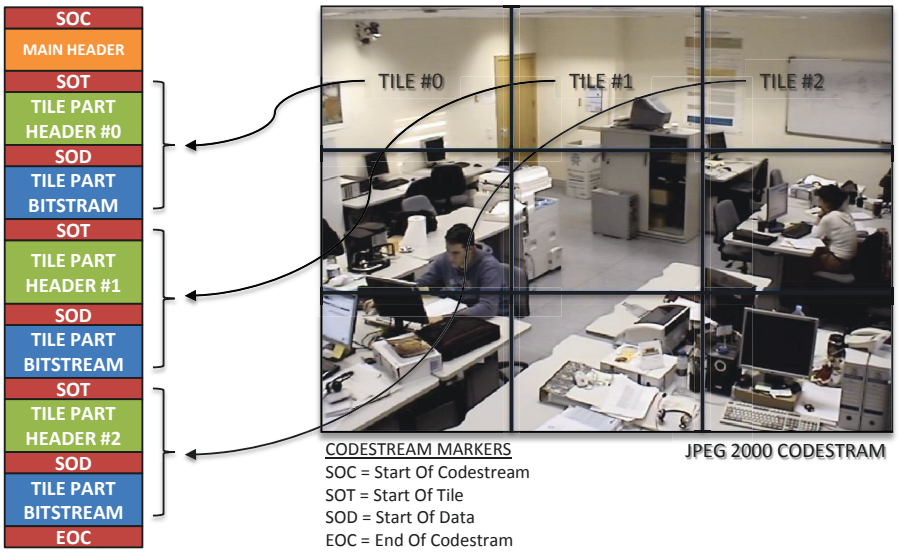


Figura 4.11: Estructura interna de una imagen comprimida con J2K

el tamaño de la imagen). Finalmente, la imagen acaba con una marca de final de código (End of Code-Stream EOC).

Podemos observar por tanto, cómo cada región de la imagen ocupa un espacio definido dentro de los datos comprimidos. Gracias a las cabeceras definidas por cada cuadrícula, y siguiendo el estándar J2K (ISO/IEC, 2000), podremos acceder a cada región de la imagen por separado.

### 4.4.3 Evaluación de la compresión de vídeo

Debido a que la compresión de vídeo implica de alguna forma alterar la fuente de vídeo original, es necesario poder evaluar la calidad de la compresión realizada. Esto se evalúa normalmente en términos de proporción de compresión, y calidad de la imagen resultante comparada con la fuente de vídeo original.

Con respecto a la calidad, o similitud en la imagen, podemos encontrar una serie de métricas estandarizadas. La métrica objetiva más común se basa en el error cuadrático medio (Mean Square Error MSE), y la relación de señal a ruido de pico (Peak to Signal Noise Ratio PSNR). El $MSE_i$ se refiere al error cuadrático acumulado entre la imagen original ($f_i$), y la imagen comprimida ($\hat{f}_i$). Por lo que el MSE es calculado pixel a pixel sobre un fotograma de $X \times Y$ píxeles 4.2.

Por otro lado, el $PSNR_i$ representa la calidad en una escala logarítmica representada en decibelios (dB). Esta es calculada a través del valor calculado de $MSE_i$ 4.3. Cuando estos métodos son aplicados para la comparación de códecs de compresión, se utilizan como una

aproximación que determina la calidad de la compresión. Esta debería estar directamente relacionada con la percepción humana. Niveles mayores de PSNR (y menor MSE) normalmente indicarán que la reconstrucción de la imagen comprimida tiene una mayor calidad.

$$MSE_i = \frac{1}{XY} \sum_{x=1}^{X} \sum_{y=1}^{Y} \left\{ f(x,y) - \hat{f}(x,y) \right\}^2 \tag{4.2}$$

$$PSNR_i = 10 \cdot \log_{10} \frac{255^2}{MSE_i} \tag{4.3}$$

Estas dos métricas (MSE y PSNR) se pueden considerar como métricas objetivas. Aunque podemos encontrar otras métricas que representan medidas de calidad subjetivas, ya que las métricas PSNR y MSE han demostrado ser inconsistentes con la percepción del ojo humano (Girod, 1993).

Estas técnicas están basadas en la visión humana y en teoría se basan en la percepción que tendría un humano sobre la imagen visualizada. Algunas de las métricas más notables son la de similitud de estructural (SSIM) (Wang et al., 2004), y la métrica de calidad visual (VQM) (Xiao et al., 2000).

La métrica SSIM se calcula sobre pequeñas regiones de las imágenes $x$ e $y$ con un mismo tamaño NxN. El cálculo de la métrica se define en 4.4, donde $\mu_x$ es la media de $x$; $\mu_y$ es la media de y; $\sigma_x^2$ es la varianza de x; $\sigma_y^2$ es la varianza de y; $cov_{xy}$ es la covarianza de y; $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ son dos variables para estabilizar la división con un denominador débil; $L$ es el rango dinámico de los píxeles (normalmente $2^{\#bitsperpixel} - 1$); y $K_1 = 0{,}01$ y $K_2 = 0{,}03$ son valores por defecto.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2cov_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{4.4}$$

El valor resultante del índice de similitud estructural será un valor decimal entre -1 y 1. El valor 1 sólo es posible alcanzarlo si las dos imágenes son exactamente iguales. Normalmente este cálculo de aplica sobre pequeñas regiones de la imágenes, con un valor por defecto de 8x8, que pueden ir desplazándose para cubrir la imagen al completo.

Por otro lado, la métrica VQM se basa en el modelo DVQ (Watson's Digital Video Quality) (Watson, 1998; Watson et al., 2001). DVQ utiliza una transformación de coseno discreta (DCT). El proceso de cálculo de esta métrica es bastante más complicado que la métrica SSIM. En la figura 4.12 se presenta el esquema de procesado que sigue este algoritmo. En

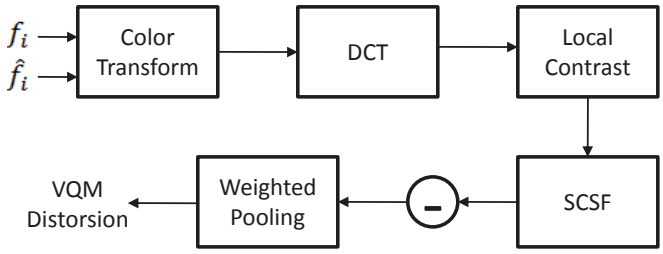


Figura 4.12: Descripción del algoritmo de evaluación de calidad subjetiva VQM.

esta métrica, valores altos indicarán poca similitud entre la imagen original y la comprimida, y valores cercanos a cero únicamente se obtendrían si la similitud es perfecta.

Para medir la reducción de información se utilizará la proporción de compresión (Compression Ratio CR). El valor de CR indicaría cuánto ha sido comprimida una imagen con respecto a la imagen original. Este valor y es fácilmente calculable con la fórmula 4.5. Cuanto más alto sea el valor de CR, mayor será la compresión.

$$CompressionRatio = \frac{UncompressedSize}{CompressedSize} \tag{4.5}$$

Figura 4.13: Secuencias de vídeo estándar utilizadas para la evaluación de compresión de vídeo

Estas métricas por tanto son de gran utilidad para evaluar la calidad de la compresión. También son necesarios un conjunto de vídeos estandarizados que permitan comparar las diferentes técnicas de compresión. Podemos encontrar múltiples secuencias diseñadas para este propósito. En este caso hemos seleccionado algunos de los más conocidos del repositorio de Xiph.org Test Media Repository (Lora, 2015). Las secuencias seleccionadas se corresponden con los vídeos 'Akiyo' y 'Hall monitor', que se muestran en la figura 4.13.

Figura 4.14: Secuencia de vídeo adicional llamada 'Surveillance'. Representa un vídeo característico de un entorno de vigilancia.

Con el fin de evaluar la compresión de vídeo también en un entorno real de videovigilancia, se propone el uso de una nueva secuencia de vídeo llamada 'Surveillance'[1]. Se puede ver un fotograma de esta secuencia de vídeo en la figura 4.14. Proviene de una cámara estática que está monitorizando la parte posterior de un edificio.

### 4.4.4 Arquitectura de compresión MIJ2K

En esta sección se detalla el método de compresión MIJ2K diseñado para optimizar la transmisión de vídeo basada en el códec J2K. Este procedimiento se aplicará al proceso de compresión y transmisión, y permitiría enviar únicamente las regiones de la imagen que han presentado algún cambio con respecto a última imagen transmitida. Para ello se utilizará la característica de cuadrícula, o tile, que incluye J2K.

Un sistema de transmisión de vídeo se compone básicamente de tres etapas. La primera está relacionada con la adquisición de las imágenes y su compresión. Seguido de la transmisión por red del resultado de la compresión. Y finalizado por la recepción y decodificación de cada fotograma en el destino.

La adquisición y transmisión en este tipo de sistemas no suele incluir ningún procesado ni procedimiento adicional. Inmediatamente después de la adquisición y transmisión de la imagen, esta es transmitida por la red. En la arquitectura MIJ2, sí se añadiría un paso adicional en el proceso de compresión. En vez de comprimir todo el fotograma, sólo se comprimen y transmiten

[1] http://www.giaa.inf.uc3m.es/miembros/alvaro/jjbase.avi

las áreas de la imagen que han sido modificadas. Comprimir y transmitir únicamente las regiones que han cambiado mejoraría el tiempo de compresión, de transmisión, y decodificación, ya que se reduciría sustancialmente la cantidad de datos a gestionar.

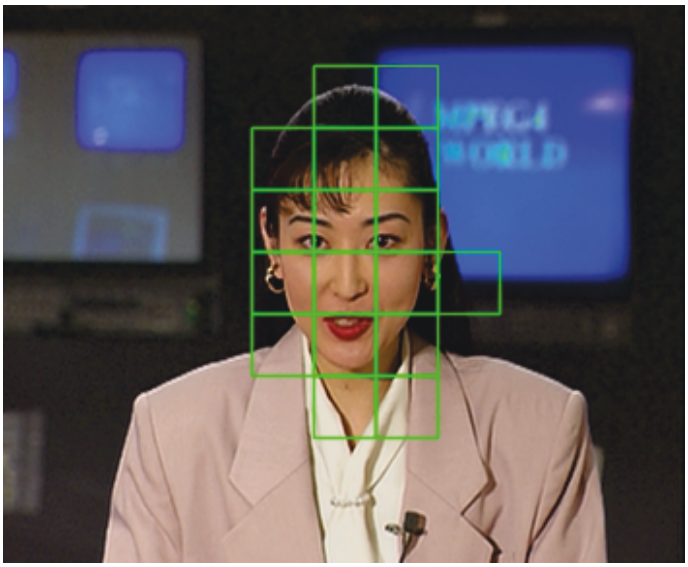

Figura 4.15: Secuencia de vídeo 'Akiyo' y las cuadrículas detectadas con cambio en el fotograma 127. En este fotograma particular podríamos ahorrar el 83 % del ancho de banda necesario para su transmisión.

Por ejemplo, en la figura 4.15 se muestra una representación de las regiones de la imagen que presentan un cambio en el fotograma 127 de la secuencia 'Akiyo'. Estas están marcadas con un rectángulo verde. En este caso, únicamente el 17 % de las cuadrículas han sido detectadas con cambio. Cuando aplicamos el sistema MIJ2K en la secuencia completa, podemos observar un ahorro del 87 % del ancho de banda necesario para transmitirla si lo comparamos con un sistema de transmisión J2K convencional.

La arquitectura diseñada para conseguir esta tarea se describe en la figura 4.16. Para detectar las regiones de la imagen que presentan cambio se utiliza un fotograma de referencia $F_i^R$ que almacena la representación del último fotograma transmitido. Cada nuevo fotograma $F_i^S$ a ser transmitido se compara cuadrícula por cuadrícula con el fotograma de referencia $F_i^R$. Este fotograma de referencia sería actualizado únicamente con las regiones que han sido marcadas con cambio. Esto permitiría representar la imagen que estaría visualizando el cliente en el otro extremo de la transmisión.

El cliente que recibe esta transmisión modificada, que no es más que un fotograma J2K que sólo contiene una parte de la imagen, tendrá que descomprimir por separado cada una de las regiones y ubicarlas en la región que les corresponda. Para ello usará un fotograma de referencia $F_i^D$ sobre el que iría aplicando todas las actualizaciones recibidas. El cliente puede ubicar fácilmente la ubicación de cada una de las cuadrículas recibidas, ya que cada una cuenta con un número que las ubica dentro de la imagen, sabiendo el tamaño de la cuadrícula y de la

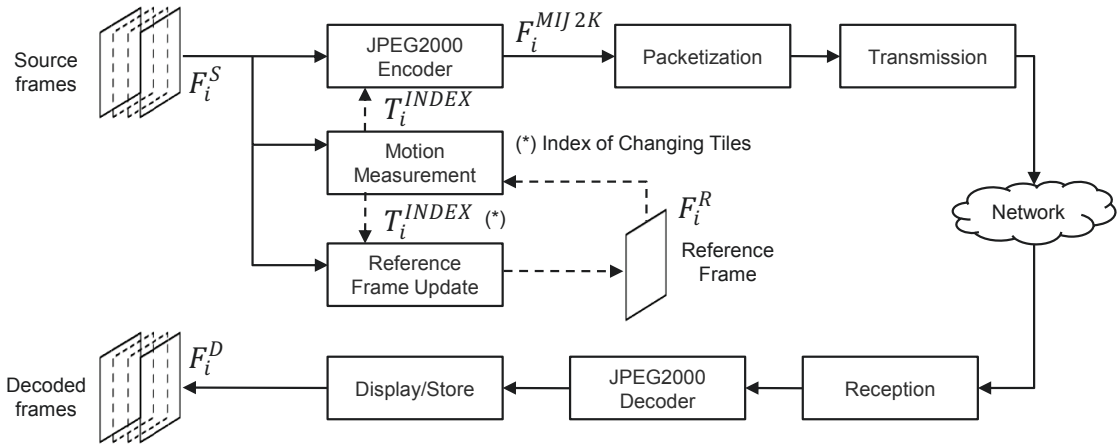


Figura 4.16: Descripción general de funcionamiento de la arquitectura MIJ2K. En esta se realiza la compresión y transmisión selectiva de las regiones de la imagen que han presentado algún cambio.

imagen (gracias a las cabeceras que contiene una secuencia J2K).

Todos los procesos ilustrados en la figura 4.16 y los detalles concretos del diseño del sistema MI2JK se explican con más detalle en las siguientes secciones.

#### 4.4.4.1 Compresión J2K

Este módulo, descrito en la figura 4.17, se encargaría de la compresión de las imágenes con el códec J2K utilizando la partición en cuadrículas. Para ello tiene que tener en cuenta que sólo es necesario comprimir las cuadrículas especificadas por el parámetro $T_i^{INDEX}$. Esto es posible realizarlo gracias a que cada cuadrícula de una imagen en J2K puede ser comprimida y descomprimida de manera independiente. Idealmente, por tanto, sólo las regiones que han recibido una actualización serían comprimidas. Esto ahorraría tiempo de compresión, mejorando en algunos casos el tiempo de procesamiento necesario para transmitir la imagen. Todas las regiones de la imagen se comprimirían con la misma calidad, en base al parámetro $bpp_i$.

A continuación se describen cada una de las partes de este módulo:

- Imagen Original $F_i^S$: Esta sería la imagen directamente captura por el digitalizador o la cámara digital, y en definitiva se trata del fotograma que se quiere transmitir. Esta imagen debería estar codificada en algún formato entendible por el compresor J2K, como puede ser una imagen RGB de $8bpp$ o $24bpp$ (dependiendo de si es a color o no).
- Calidad de Compresión $bpp_i$: Este parámetro está relacionado con la proporción de compresión requerida. Este parámetro se suele expresar en términos de número de bits a utilizar por cada pixel comprimido de la imagen J2K. Esto es lo mismo que bits por píxel ($bpp$).

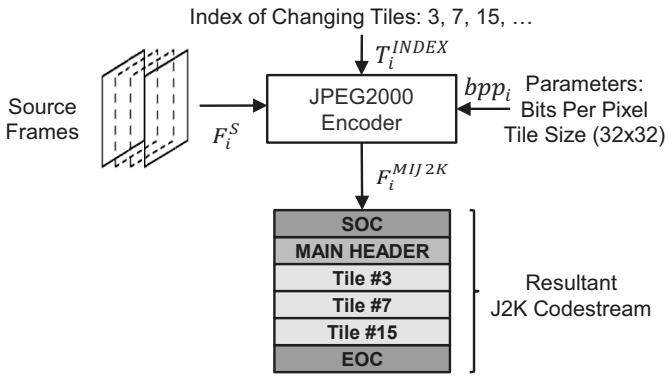


Figura 4.17: Compresor de imágenes en J2K que tiene en cuenta diferentes parámetros de entrada, como el tamaño de la cuadrícula, el índice de los elementos que presentan cambios, la calidad de compresión, etc.

- Tamaño de Cuadrícula $T_{SIZE}$: Permite definir el tamaño de la cuadrícula a utilizar en la compresión de la imagen. El tamaño de las cuadrículas es variable. En teoría, un tamaño menor podría proporcionar un mejor ajuste a los objetos que se mueven en la escena. Esto se puede observar en la figura 4.18, donde un jugador de fútbol es detectado teniendo en cuenta diferentes tamaños de cuadrícula. Se puede observar que la cuadrícula más pequeña se ajusta mucho mejor a la forma del jugador que se está moviendo. Pero tras un proceso de optimización realizado en un trabajo adicional descrito en (Luis Bustamante et al., 2009), se ha comprobado que uno de los tamaños que mejor resultados proporciona es el de $32 \times 32$ píxeles, por lo que será el valor a utilizar por defecto.

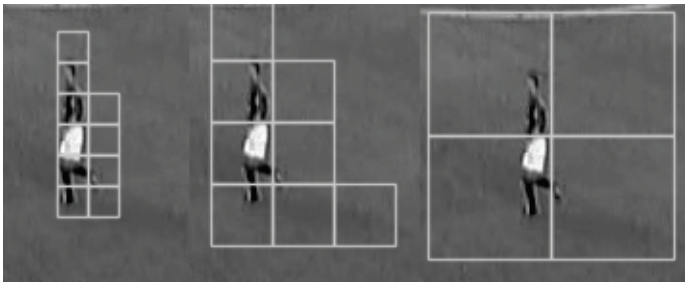

Figura 4.18: Ejemplo de cuadrículas de diferentes tamaños para la detección de movimiento. De izquierda a derecha podemos ver tamaños de $16 \times 16$, $32 \times 32$ y $64 \times 64$ píxeles.

- Índice de cuadrículas con cambio $T_i^{INDEX}$: Esta entrada es proporcionada por el módulo de detección de movimiento que se describirá más adelante. Esta indicaría el índice de cada una de las cuadrículas que han sido detectados con cambio, y por tanto que deben ser comprimidas. Esto evitaría tener que comprimir toda la imagen, pudiendo ganar algo de tiempo en el proceso de compresión.

- Imagen Comprimida $F_i^{MIJ2K}$: Este es el resultado de la compresión. Este contendrá únicamente la compresión de las regiones detectadas con cambio que se indicaron con

en el parámetro de entrada $T_i^{INDEX}$. El resultado de la compresión es procesado por el sistema de empaquetado que adecuará la información al protocolo de red utilizado.

Este proceso podría resumirse en la función 4.6, donde J2K representaría el sistema de compresión. Este contaría con los parámetros de imagen original $F_i^S$, índice de las cuadrículas a comprimir $T_i^{INDEX}$, calidad de la compresión $bpp_i$, y el tamaño de las cuadrículas $T_{SIZE}$ con un tamaño por defecto de $32 \times 32$ píxeles.

$$F_i^{MIJ2K} = J2K\left(F_i^S, bpp_i, T_i^{INDEX}, T_{SIZE} = 32 \times 32\right) \tag{4.6}$$

#### 4.4.4.2 Detección de movimiento

Este módulo sería el encargado de realizar la detección de movimiento entre dos fotogramas consecutivos, y por tanto proporcionar el índice de las cuadrículas a comprimir. El algoritmo seleccionado para realizar esta tarea ha sido seleccionado teniendo en cuenta que tiene que tener una baja complejidad computacional. Esto permitirá no interferir demasiado en el proceso de compresión ni añadir retardos innecesarios.

Este algoritmo tomaría como entrada dos fotogramas diferentes, $F_i^S$ y $F_i^R$, que representan el fotograma origen y el de referencia usado en la comparación. También usaría como parámetro el tamaño de la cuadrícula $T_{SIZE}$ que queremos utilizar en el proceso de compresión. Esta representación puede observarse en la figura 4.19.

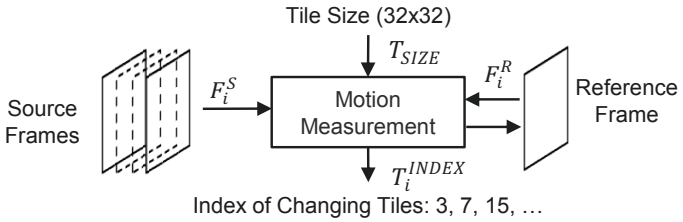


Figura 4.19: Descripción de entrada y salida del algoritmo de detección de movimiento.

La información de entrada y salida se describe con más detalle a continuación:

- Imágenes de Entrada $F_i^S$ and $F_i^R$: $F_i^S$ sería la misma imagen proporcionada al módulo de compresión. La imagen directamente capturada de la digitalizadora o la cámara digital. La imagen $F_i^R$ sería la imagen de referencia que se utiliza para comparar y detectar los movimientos de la imagen de entrada $F_i^S$.

- Tamaño de Cuadrícula $T_{SIZE}$: Como el sistema de compresión se basará en el concepto de las cuadrículas, el algoritmo de detección tendrá que adaptarse también al tamaño

seleccionado. Debe coincidir con el tamaño proporcionado al algoritmo de compresión. Mantendrá el valor por defecto de $32 \times 32$ píxeles.

- Índice de cuadrículas con cambio $T_i^{INDEX}$: Como se ha visto en la sección del módulo de compresión, el módulo de detección será el encargado de proporcionar los índices de las cuadrículas que han cambiado. El número de índices suministrado para cada fotograma será completamente variable en función de la imagen de entrada y la de referencia.

$$T_i^{INDEX} = Motion\left(F_i^S, F_i^R, T_{SIZE} = 32 \times 32\right) \tag{4.7}$$

De nuevo podemos resumir el funcionamiento de este módulo según la función 4.7. En este caso especificaremos con más detalle el algoritmo de detección de movimiento aplicado. En la figura 4.20 podemos observar un esquema del general del funcionamiento del algoritmo. Desarrollaremos a continuación además cada uno de los procesos involucrados.

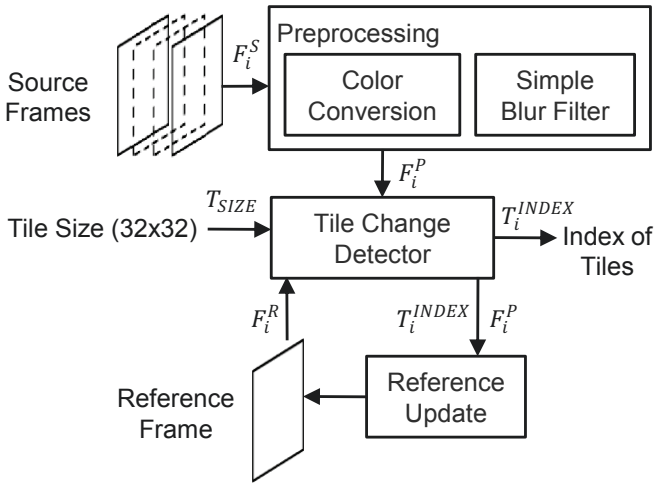


Figura 4.20: Esquema de funcionamiento del sistema de detección de movimiento diseñado para el sistema de compresión MIJ2K.

- Preprocesado: Esta tarea prepara la imagen de origen $F_i^S$ para el proceso de detección de movimiento. Para ello se extrae la información de intensidad del espacio de color YUV de la imagen. Para ello se aplica la conversión especificada en 4.8 para las imágenes RGB de entrada. De esta podemos trabajar con una representación de la imagen más sencilla, pudiendo aplicar algoritmos de análisis más rápidamente.

$$Y = 0{,}2999R + 0{,}587G + 0{,}114B \tag{4.8}$$

Una vez que la imagen ha sido convertida correctamente a escala de grises, se aplicaría un algoritmo de Blur (que hace más borrosa la imagen) que permita reducir el excesivo detalle y ruido que suelen presentar las imágenes de las cámaras de vigilancia.

La figura 4.21 permite ilustrar el efecto de aplicar un algoritmo de Blur para la reducción de ruido presente en la imagen. Las imágenes a y b son dos fotogramas consecutivos de la secuencia 'Surveillance' sobre los que se ha extraído la componente Y. Las imágenes c y d representan la diferencia absoluta entre las imágenes $n$ y $n+1$. La imagen c presenta la diferencia absoluta cuando no se aplica el algoritmo de Blur a las imágenes a comprar. La imagen d presentaría lo mismo pero en este caso las imágenes han pasado por el algoritmo de Blur. A simple vista parece no haber mucha diferencia entre ambas imágenes, pero si realizamos una binarización de las imágenes podemos ver la cantidad de ruido que aparece en la imagen que no ha sido tratada con Blur. Además este ruido ha sido generado por elementos estáticos del entorno, que no estamos interesados en volver a transmitir. Las imágenes a las que se les ha aplicado el algoritmo de Blur presentan una binarización mucho más limpia, descartando prácticamente toda la información ruidosa.

Se podrían aplicar otras técnicas como erosión y dilatación, pero son computacionalmente más costosas. Además, es más difícil ajustar un elemento estructurante apropiado para prevenir la pérdida de demasiada información (Van Droogenbroeck and Talbot, 1996).

Por lo tanto, el fotograma preprocesado $F_i^P$, será básicamente una imagen en escala de grises sobre la que se ha aplicado un filtro Blur, tal y como se describe en la función 4.9.

$$F_i^P = Blur\left(GrayScale\left(F_i^S\right)\right) \tag{4.9}$$

- Detección de movimiento de cuadrículas: Este módulo, ilustrado en la figura 4.20, se encargará de detectar las cuadrículas que presentan algún tipo de movimiento. El método empleado aquí debe ser relativamente sencillo para evitar añadir retardos innecesarios en el proceso de captura, compresión y transmisión.

  Este módulo utilizaría la imagen preprocesada $F_i^P$ que sería comparada con la imagen de referencia $F_i^R$. Para ello se compararán las imágenes siguiendo la división en cuadricula que hemos descrito. Para cada región de la imagen se detecta si se ha producido suficiente movimiento para así decidir si ésta será transmitida o no. En la figura 4.22 podemos observar cómo se realizaría este proceso, cuyo funcionamiento detallaremos a continuación.

- Diferencia Absoluta $ABS_{i[x]}$: Cada región de la imagen de origen y de referencia pasan por un proceso de cuantificación para detectar los cambios absolutos entre cada una de las regiones. La complejidad computacional de este proceso es muy baja, y es muy útil para detectar cambios de forma objetiva entre las zonas. El resultado de este proceso genera

(a) Fotograma $n$.

(b) Fotograma $n+1$.

(c) Diferencia Absoluta. Sin Blur.

(d) Diferencia Absoluta. Con Blur

(e) Binarización. Sin Blur.

(f) Binarización. Con Blur.

Figura 4.21: Efectos de la aplicación de un algoritmo de Blur en la reducción de ruido de la imagen.

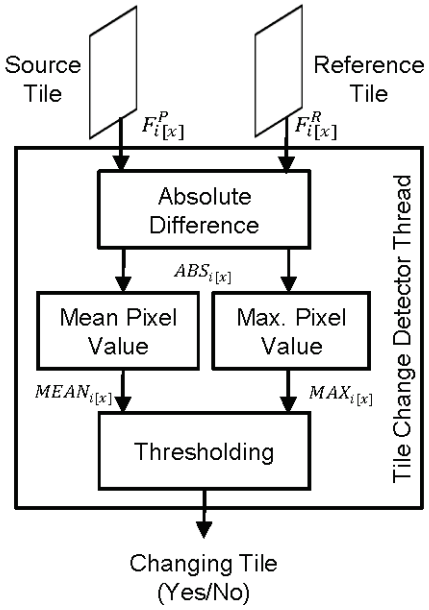


Figura 4.22: Sistema de detección de movimiento de regiones implementado en la arquitectura MIJ2K

una imagen en blanco y negro donde los píxeles blancos representan cambio. Este proceso se describe en 4.10, donde $F^P_{i[x]}$ y $F^R_{i[x]}$ representan la región $x$ del fotograma $i$ en la imagen preprocesada $P$ y de referencia $R$. Por otro lado, $ABS_{i[x]}$ representa la diferencia absoluta entre las regiones. En este caso, tanto $F^P_{i[x]}$ como $F^R_{i[x]}$ son dos matrices de $u \times v$ que contienen los valores de los píxeles de las imágenes.

$$ABS_{i[x]} = \left| F^P_{i[x]} - F^R_{i[x]} \right| \tag{4.10}$$

La imagen resultante $ABS_{i[x]}$ de la comparación de una zona debe analizarse para detectar la cantidad de cambio que presenta. Esta puede medirse aplicando algún criterio concreto. En este caso se proponen dos métodos eficientes que trabajarán en conjunto.

- Valor medio $MEAN_{i[x]}$: Esta sería la primera medida, y representa la media del valor de la matriz resultante del cálculo de la diferencia absoluta $ABS_{i[x]}$, tal y como se describe en la ecuación 4.11. Servirá para detectar cambios ligeros y homogéneos que a lo mejor de forma individual pasarían desapercibidos.

$$MEAN_{i[x]} = \frac{\sum_{s=0}^{u-1} \sum_{t=0}^{v-1} ABS_{i[x](s,t)}}{u \times v} \tag{4.11}$$

- Valor máximo $MAX_{i[x]}$: El segundo criterio consistirá en detectar el valor más alto entre todos los píxeles, como se describe en 4.12. Este indicador será útil para detectar grandes cambios en pequeñas regiones de las zonas, y que podrían pasar desapercibidos utilizando únicamente el criterio de la media de cambio.

$$\begin{aligned} \forall\, s,t \;/\; s \geq 0 \;\wedge\; s < u,\; t \geq 0 \;\wedge\; t < v \\ MAX_{i[x]} = ABS_{i[x]_{(s,t)}} \Leftrightarrow \\ \neg\exists\; ABS_{i[x]_{(g,h)}} > ABS_{i[x]_{(s,t)}} \end{aligned} \tag{4.12}$$

  En conjunto, los criterios de $MEAN_{i[x]}$ y $MAX_{i[x]}$ podrán detectar prácticamente todo tipo de movimientos. Desde cambios ligeros y uniformes, a cambios importantes que se presentan de forma puntual. Ambas métricas son fáciles de implementar, y además proporcionan una baja complejidad. Esta complejidad será analizada en los posteriores experimentos donde comprobaremos el retardo introducido.

- Umbralización: Los indicadores $MEAN_{i[x]}$ y $MAX_{i[x]}$ nos permitirán decidir cuando una determinada región o zona debe ser transmitida. Para ello necesitaremos utilizar algún umbral que nos permita discernir cuando una región se considera que presenta cambio. Para ello utilizaremos los valores resultados de la optimización descritos en el trabajo (Luis Bustamante et al., 2009). A modo de resumen, podemos considerar un umbral de 2.0 para el indicador de $MEAN_{i[x]}$ y un umbral de 15 para $MAX_{i[x]}$. Este ajuste está optimizado para el tamaño de la cuadrícula de $32 \times 32$ píxeles.

- Actualización de imagen de referencia: Esta parte se encargaría de la actualización de la imagen de referencia $F_i^R$. La primera imagen de referencia, $F_0^R$ consistirá en una imagen negra, y que será actualizada región a región por los diferentes imágenes preprocesadas $F_i^P$. Lógicamente, el primer fotograma $F_0^P$ provocará la actualización de tosas las regiones de la imagen de referencia. Nótese que la imagen de referencia se actualiza directamente de la imagen que ha pasado por el algoritmo de Blur generada por el módulo de preprocesado. Esto evitará tener que hacer un preprocesado continuo a la imagen de referencial, lo que mejorará los tiempos de procesado.

#### 4.4.4.3 Partición en paquetes RTP

Este módulo entraría en operación tras el proceso de detección de movimiento y compresión. Aquí se cogería la salida de la imagen comprimida $F_i^{MIJ2K}$ con las regiones donde se ha presentado movimiento. Este sistema se encargaría de procesar los datos binarios para realizar una transmisión eficiente a través de la red. Esta transmisión se haría siguiendo el estándar

contemplado en el RFC 5371 (Futemma et al., 2008). La implementación está más detallada en el trabajo publicado en (Luis and Patricio, 2009).

Básicamente consistiría en coger la imagen comprimida y generar un conjunto de paquetes RTP que representen las cabeceras y las regiones comprimidas. Para esto, cada paquete RTP usa una cabecera de contenido específicamente definida para transmitir secuencias de vídeo basadas en J2K, tal y como se define en el estándar RFC 5371 (Futemma et al., 2008).

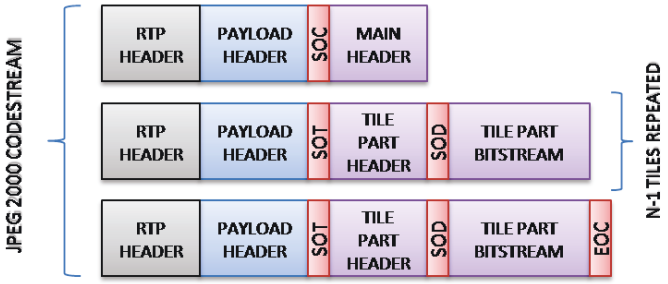


Figura 4.23: Transmisión de las imágenes J2K a través del protocolo de transmisión en tiempo real RTP (Real-time Transport Protocol).

La partición de las imágenes se puede observar en la figura 4.23. Cada parte de la imagen se transmitirá en un paquete RTP independiente. Esto es, el primer paquete contendrá la cabecera, seguido de cada región que representa una parte de la imagen. Algunos de los beneficios de este modo de transmisión se detallan en (Luis and Patricio, 2009), y a modo de resumen podemos destacar los siguientes.

- Procesamiento en paralelo: No es necesario comprimir la imagen completa antes de enviarla. Podemos comprimir una región de la imagen de manera independiente y transmitirla según esté disponible. Esto nos permite paralelizar la tarea de compresión y transmisión, disminuyendo la latencia. Lo mismo ocurriría de cara al cliente que se encuentre recibiendo esta información, ya que podría decodificar y visualizar cada región de manera independiente mucho antes de que llegue la representación completa de la imagen.

- Tolerancia a fallos: Si se pierde un paquete durante la transmisión por red, esta pérdida afectaría únicamente a la región que representaba, por lo que no se perdería la imagen al completo.

- Reemplazo de la cabecera principal: La cabecera principal de una imagen J2K es la parte más importante de la imagen. Sin esta es imposible decodificar la imagen. Por este motivo es importante gestionar la pérdida del paquete que representa la cabecera. Esto se puede solucionar proporcionando un identificador de cabecera dentro de cada paquete RTP. Cada identificador de cabecera representaría a un tamaño de imagen, número de componentes, bits por píxel, tamaño de la cuadrícula, etc. Como esta información no

suele variar durante la transmisión, esta información se puede almacenar en la recepción. Posteriormente, cada paquete RTP con información de la imagen haría referencia al identificador de cabecera que necesita para su decodificación. Esto hace el sistema más tolerante a pérdidas de la cabecera principal. Inclusive permitiría el ahorro del paquete de la cabecera principal una vez esta ha sido recibida.

#### 4.4.4.4 Transmisión y Recepción

Estos subsistemas estarían involucrados en la transmisión y recepción de los paquetes RTP. Los paquetes RTP deben ser transmitidos sobre un protocolo de transporte que permitan definir la fuente y destino de la transmisión. Para esta tarea usaremos el protocolo de transporte UDP (User Datagram Protocol). Este protocolo nos permitiría cumplir los requisitos de transmisión en tiempo real, ya que evita la sobrecarga introducida por otros protocolos como el TCP. Además, el protocolo UDP no está orientado a conexión, lo que nos permitiría hacer multi-difusión (multicast) a diferentes clientes.

Es importante que el canal de comunicación empleado sea relativamente fiable y proporcione suficiente ancho de banda para la transmisión de vídeo en tiempo real. Esto no suele suponer un problema hoy día, ya que inclusive las conexiones de red a Internet están mejorando notablemente en los últimos años. Es fácil encontrar hoy día conexiones de ancho de banda de 100Mbps en los hogares.

Las pruebas que nosotros realizaremos se harán sobre una red de área local con un enlace de 100Mbps. Seguramente utilizando redes más avanzadas como Gigabit Ethernet o fibra óptica, nos permitirían mejorar el resultado de las pruebas y el número de canales de vídeo soportados.

#### 4.4.4.5 Decodificador J2K

El módulo de decodificación dentro de la arquitectura MI2JK se encargaría de la decodificación de las regiones de la imagen tan pronto como sean recibidas. Como hemos visto en la partición de los paquetes RTP, cada parte de la imagen es transmitida de manera independiente. Esto nos permite ir decodificando y visualizando cada región recibida mientras continuamos con la recepción del resto.

La figura 4.24 mostraría este proceso. El primer paquete recibido consistiría en la cabecera de la imagen JPGE2000 que nos permitiría decodificar el resto de la imagen. El resto de paquetes correspondería a cada región de la imagen, que se irían decodificando y visualizando según son recibidas. Es importante destacar aquí la posibilidad de paralelizar la decodificación en múltiples hilos de ejecución. De esta forma podemos contar con un conjunto de hilos preparados para decodificar las regiones que se van recibiendo. Esto mejoraría notablemente los tiempos de decodificación y visualización en dispositivos que soporten múltiples hilos de ejecución.

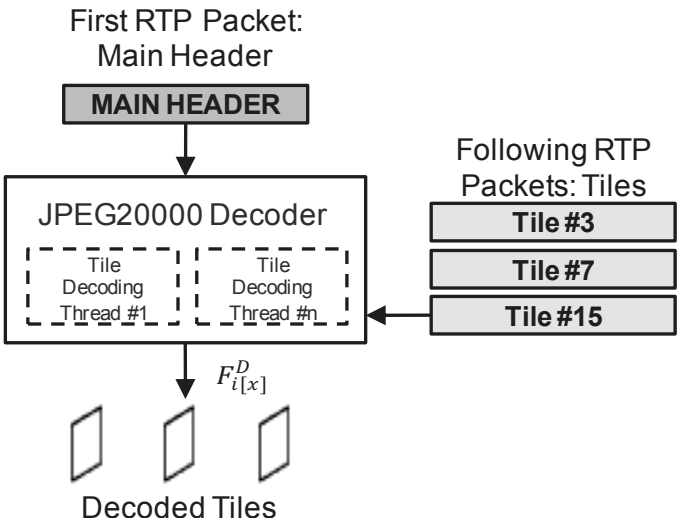


Figura 4.24: Decodificación de las secuencias J2K. Esta tarea puede paralelizarse en diferentes hilos de ejecución para cada región de la imagen.

#### 4.4.4.6 Visualización/Almacenamiento de la imagen

El sistema de visualización o almacenamiento de la secuencia de vídeo es bastante intuitivo. Cada región $F^D_{i[x]}$ de la imagen recibida del proceso de descompresión se reemplazaría sobre el fotograma $F^D_i$. De esta forma estaríamos actualizando la imagen con las regiones que han presentado cambios dentro de la imagen. Esta imagen se podría utilizar para su visualización en un monitor o para almacenarla en algún formato de vídeo.

### 4.4.5 Evaluación del sistema MIJ2K

En esta sección detallaremos las pruebas realizadas sobre la arquitectura MIJ2K propuesta para evaluar las posibles mejoras y ventajas que podemos esperar. Evaluaremos esta arquitectura en los siguientes términos:

- Calidad del vídeo: El sistema MIJ2K añade un sistema de compensación de movimiento a un sistema de transmisión basado en J2K. Por lo tanto queremos evaluar la mejora que presentaría contra un sistema nativo de este tipo. Además, lo evaluaremos contra otros sistemas como MJPEG o H.264-Intra.
- Complejidad computacional: No tendría sentido introducir un mecanismo de compensación de movimiento si esto nos va a suponer incrementar la latencia general del sistema. Por este motivo evaluaremos cómo afecta este método al proceso de transmisión de vídeo, haciendo hincapié en los posibles retardos introducidos.

- Latencia: En este experimento evaluaremos la latencia de la implementación de referencia que hemos desarrollado. Utilizaremos el entorno real que hemos descrito al comienzo de la sección del entorno multi-cámara para evaluar la latencia global en un sistema real. Para ello tendremos en cuenta el proceso completo, desde la adquisición, compresión, transmisión, decodificación y visualización. Esto nos permitirá evaluar su adecuación para un sistema real de vigilancia.

Para el proceso de evaluación de calidad utilizaremos el conjunto de vídeos estándar definidos en la sección 4.4.3. Estas son las secuencias 'Akiyo', 'Hall Monitor' y 'Surveillance'. Los vídeos se encuentran codificadas en un formato sin compresión RGB de 24 bits. Podemos encontrar una descripción de los mismos en la tabla 4.7.

| Sequence | Resolution | Length | Original Size |
|---|---|---|---|
| Akiyo | 352x288 | 251f, 10s | 72.8 MB |
| Hall Monitor | 352x288 | 251f, 10s | 72.8 MB |
| Surveillance | $640 \times 480$ | 704f, 28s | 622.3 MB |

Tabla 4.7: Descripción de las secuencias de vídeo seleccionadas para evaluar el sistema de compresión y transmisión MIJ2K.

#### 4.4.5.1 Comparación con un sistema J2K convencional

En este experimento queremos evaluar la mejora introducida sobre un sistema basado en el J2K convencional. Para ello evaluaremos la calidad del vídeo comprimido en cada una de las secuencias de prueba, que serán comprimidas tanto con el sistema MIJ2K, como con el estándar J2K. Todas las secuencias han sido comprimidas aproximadamente con la misma proporción de compresión (CR). El sistema J2K será configurado con una tasa de bits constante, y se comprimirá sin división en cuadrículas (lo que le debería aportarle mejoras significativas). El sistema MJ2K también comprimirá con una tasa de bits constante, pero en este caso contará con la división en cuadrícula y la técnica de compensación de movimiento descrita.

Los parámetros utilizados para la compresión MIJ2K corresponden a los valores por defecto descritos en la sección anterior. Estos son, un tamaño de cuadrícula $T_{SIZE}$ de $32 \times 32$, 2.0 para el umbral $MEAN_{i[x]}$, y 15 para el umbral $MAX_{i[x]}$. Estos valores han sido extraídos del proceso de optimización realizados en el trabajo (Luis Bustamante et al., 2009). El software de compresión/descompresión utilizado para estas pruebas ha sido el proporcionado por el SDK Kakadu J2K. El resumen de los resultados obtenidos en estos experimentos los podemos encontrar en la tabla 4.8.

Las figuras 4.25, 4.26 y 4.27 nos permiten observar que el sistema MIJ2K mejora sustancialmente la calidad de compresión cuando las secuencias de vídeo se comprimen con la misma

| Sequence | MIJ2K Size | J2K Size | MIJ2K CR | J2K CR | MIJ2K Avg. PSNR | J2K Avg. PSNR |
|---|---|---|---|---|---|---|
| Akiyo | 1265KB | 1201KB | 59:1 | 62:1 | 41.2525dB | 36.6815dB |
| Hall Monitor | 2070KB | 2138KB | 36.01:1 | 34.86:1 | 39.9168dB | 34.9348dB |
| Surveillance | 2534KB | 2556KB | 251.46:1 | 249.302:1 | 39.5379dB | 30.9377dB |

Tabla 4.8: Resultado de la evaluación de calidad de compresión entre MIJ2K y un sistema de compresión tradicional basado en J2K.

proporción de compresión. En estas figuras se puede observar el valor de PSNR obtenido para cada uno de los fotogramas de cada secuencia. Claramente, el valor de PSNR obtenido por el sistema MIJ2K está por encima de del valor obtenido por el códec J2K. Recordemos además que el sistema J2K estaría utilizando una compresión sin cuadrículas, lo que le debería dar ventajas en la compresión. Aún así esto no es suficiente para superar al MIJ2K.

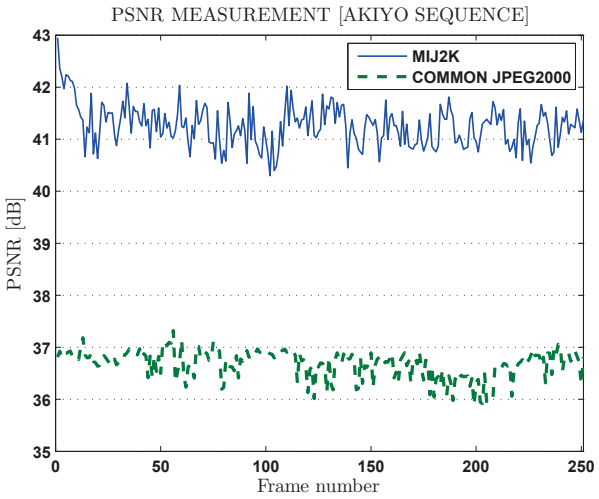


Figura 4.25: Calidad de compresión de vídeo de la secuencia 'Akiyo'.

El aumento extra de PSNR obtenido con el sistema MIJ2K en relación con el sistema J2K parece ser dependiente de la secuencia comprimida, y especialmente de la resolución de la imagen. 'Akiyo' y 'Hall Monitor' comparten la misma resolución, y en ambos casos la diferencia de calidad es de alrededor 5 dB en favor de MIJ2K si nos fijamos en la media de PSNR de tabla de resultados 4.8. Por otro lado, la secuencia 'Surveillance', que tiene mejor resolución, permite incrementar la posibilidad de aparición de áreas sin movimiento dentro de la imagen. Esto permite reutilizar múltiples regiones de la imagen, para en este caso en particular aventajar al tradicional J2K en unos 9 dB. Por lo tanto, podríamos esperar rendimientos similares en imágenes de mayor resolución.

Para hacernos una idea del significado de estos resultados, podemos tener en cuenta que el ojo humano es capaz de detectar variaciones en la calidad de una imagen con cambios de aproximadamente 0.5 dB. En este caso estaríamos incrementando este valor entre 10 y 18 veces para las secuencias analizadas. Inclusive podemos proporcionar una comparación de dos

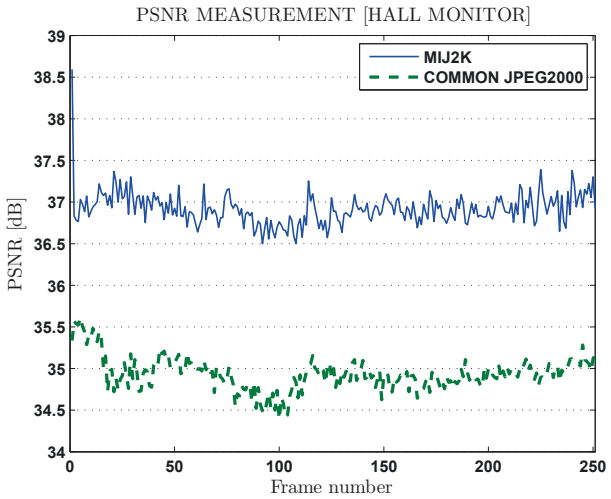


Figura 4.26: Calidad de compresión de vídeo de la secuencia 'Hall Monitor'.

imágenes que han sido comprimidas con J2K y MIJ2K 4.28. Podemos observar una mejor definición de los contornos en la imagen con MIJ2K, mientras que la imagen con J2K aparece más borrosa.

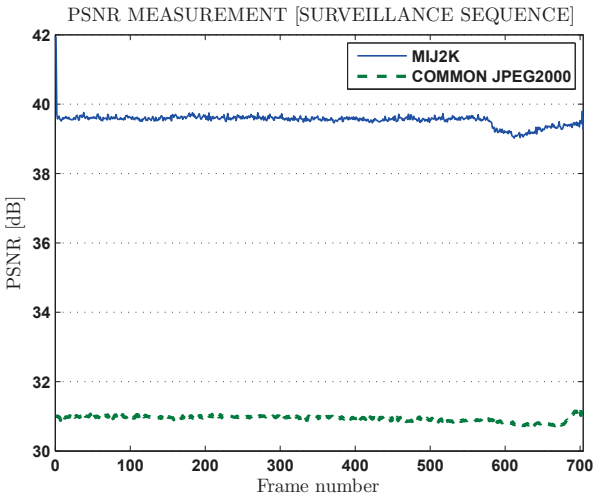


Figura 4.27: Calidad de compresión de vídeo de la secuencia 'Surveillance'.

Podemos observar cómo el tamaño de compresión de cada fotograma varía dependiendo del movimiento detectado en cada escena. La figura 4.29 nos permitiría visualizar la variación de tamaño de cada fotograma para la secuencia 'Akiyo'. En este caso, podemos ver un pico inicial en la transferencia, que sería causado por la transmisión inicial de la imagen completa. Después de este envío se estarían reutilizando las regiones de la imagen sin movimiento, lo que permite reducir sustancialmente el tamaño necesario para cada fotograma. Podemos observar además cómo el sistema J2K estaría generando un tamaño constante para cada fotograma, al

Figura 4.28: Diferencia de calidad entre un fotograma comprimido con J2K (a la izquierda), y otro comprimido con MIJ2K (a la derecha) para la misma proporción de compresión.

no usar ninguna técnica de compensación de movimiento.

Sin embargo podemos observar cómo algunos fotogramas comprimidos con MIJ2K ocupan más espacio que aquellos comprimidos con J2K. Esto tiene dos motivos fundamentales. El primero consiste en la calidad de compresión utilizada para el MIJ2K. Para llegar a obtener la misma proporción de compresión que la obtenida con J2K, ha sido necesario aumentar la calidad de compresión. De haber utilizado la misma calidad de compresión que el J2K, habríamos obtenido una proporción de compresión mucho mayor, pero no nos habría permitido comparar las calidades. El segundo motivo consistiría en el uso de cuadrículas para la compresión, lo que reduce la eficiencia del algoritmo de compresión.

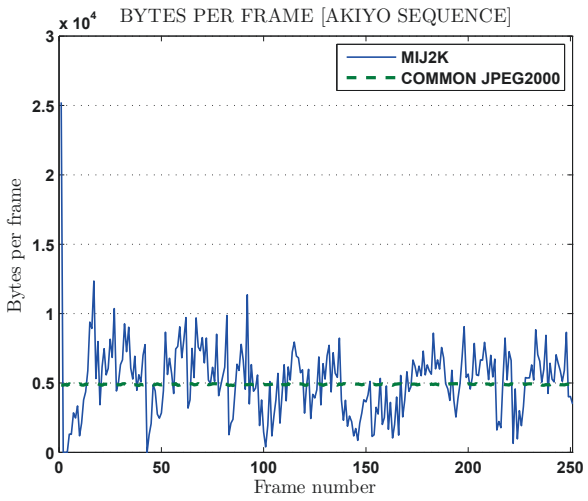


Figura 4.29: Bytes por cada fotograma utilizado en la compresión de la secuencia 'Akiyo'.

De forma general podemos concluir que el uso del sistema MIJ2K mejoraría sustancialmente al sistema tradicional de compresión de vídeo basado en el códec J2K. Esto nos permitiría

salvar ancho de banda para obtener la misma calidad de imagen. O al contrario, obtener una mejor calidad en la imagen para el mismo ancho de banda.

#### 4.4.5.2 Retardo en la detección de movimiento

En esta prueba evaluaremos de manera experimental la complejidad del algoritmo de movimiento integrado en el sistema MIJ2K. Para ello mediremos el tiempo total necesario empleado por el módulo de detección de movimiento. Esto incluye todos los procesos realizados por este módulo, desde las técnicas de preprocesado de las imágenes de entrada $F_i^S$, hasta la actualización del fotograma de referencia $F_i^R$.

La gran mayoría de las operaciones que realiza este módulo, como la conversión en escala de grises, la aplicación del algoritmo de Blur, la diferencia absoluta entre el fotograma de entrada y el de referencia, y el cálculo de las métricas de movimiento, se han desarrollado en C++ utilizando la librería OpenCV. En este caso, aún no se ha implementado la paralelización de estas tareas (mediante la división de la imagen), por lo que el proceso aún tendría margen de mejora.

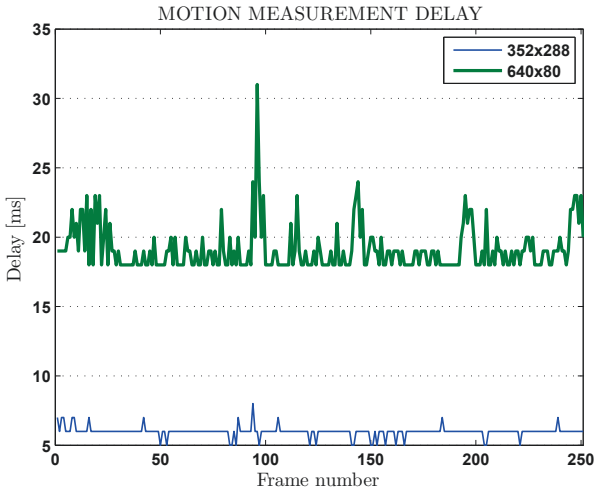


Figura 4.30: Tiempo empleado por el módulo de detección de movimiento empleado en la arquitectura MIJ2K.

El experimento se ha realizado sobre un portátil con un procesador Intel Centrino2 Duo a 1.66 Ghz. Cuenta con 2GB de DDR2 a 533Mhz, y un sistema operativo Windows Vista de 64 bits. El tamaño de las cuadrículas sigue siendo de $32 \times 32$ píxeles, para las dos resoluciones que analizaremos de $352 \times 288$ y $640 \times 480$. La medición del tiempo empleado por el módulo se ha realizado utilizando la API de Windows QueryPerformanceFrequency.

Podemos contemplar en la figura 4.30 el resultado de este experimento, donde se puede observar el tiempo empleado medido en milisegundos para cada una de las resoluciones. Podemos

ver un tiempo de procesado de alrededor de 6ms para los fotogramas de tamaño $352 \times 288$. Por otro lado, en los fotogramas de $640 \times 480$, nos encontraremos con una media de procesado de 19 ms. Teniendo en cuenta que una cámara tradicional es capaz de proporcionar una media de 25 fotogramas por segundo, o lo que es lo mismo, un fotograma cada 40 ms, podemos considerar que estos resultados son aceptables. Esto nos deja un margen de tiempo suficiente entre fotograma y fotograma para aplicar el algoritmo de detección de movimiento, lo que no acarrearía retardos incrementales en la cadena de procesado. Aún así, este proceso podría ser optimizado mediante la paralelización del procesado de las diferentes regiones de la imagen, que sería especialmente útil en imágenes de mayor resolución.

Además hay que considerar que este proceso permitirá reducir el tiempo de compresión, ya que estaríamos evitando la compresión completa de cada fotograma. Por tanto, cuando combinemos el sistema de detección de movimiento con la compresión, es de esperar que al final se reduzcan los tiempos totales hasta generar la imagen comprimida.

#### 4.4.5.3 Latencia total

En este experimento evaluaremos la latencia total del sistema, esto es, el retardo con el que el cliente visualiza la imagen. Esto implica medir el tiempo empleado en toda la cadena de procesado, desde la adquisición de la imagen, compresión, transmisión, descompresión, y visualización. En este caso estamos evaluando algo más que la arquitectura MIJ2K, ya que estamos introduciendo elementos externos como la transmisión por red. Pero nos permitirá obtener una idea del rendimiento esperado, así como su adecuación a los entornos de vigilancia con los que estamos trabajando.

Para medir la latencia se ha utilizado el siguiente equipamiento:

- Servidor: Ordenador ejecutando Windows XP Service Pack 2. Cuenta con un Intel Core 2 Duo a 2.0Ghz, con 2GB de DDR2 a 533 Mhz. Utiliza tarjetas digitalizadoras Matrox Morphis, que permiten la adquisición y compresión de las secuencias J2K. El software de transmisión se basa en el estándar RFC 5371 que hemos descrito, y que hemos desarrollado en C++ para la ocasión.
- Red: Conexión de red de área local a 100Mbps. Realizando una transmisión multi-destino desde el servidor. Tanto el cliente como el servidor están conectados al mismo conmutador de la marca CISCO.
- Cliente: El hardware del cliente es el mismo que el descrito para la medición de tiempo del módulo de detección de movimiento. El cliente ejecuta un software que permite recibir, decodificar, y descomprimir las imágenes J2K. Este software también ha sido desarrollado en C++, usando en este caso la librería Kakadu para la descompresión de las imágenes J2K.

Una forma práctica de medir la latencia total del sistema podría consistir en visualizar un reloj con precisión de milisegundos en la pantalla del cliente, enfocar una de las cámaras hacia la pantalla, y a la vez visualizar la secuencia de vídeo recibida. En la pantalla se mostrarían los dos relojes. El generado por el cliente, y el recibido a través del sistema de transmisión de vídeo. Comparando los dos relojes podremos medir de una forma muy precisa el tiempo total que estamos empleando en toda la cadena de procesado.

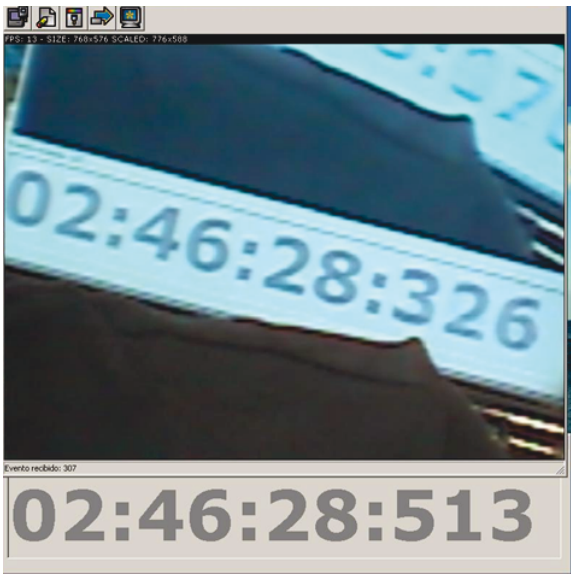


Figura 4.31: Experimento de para medir la latencia total del sistema de transmisión de vídeo. Podemos observar una latencia de 187 ms para todo el proceso de adquisición, compresión, transmisión, y decodificación.

Podemos ver una captura de pantalla de este experimento en la figura 4.31. En esta podemos visualizar los dos relojes mencionados, y podemos comprobar que la diferencia de tiempo entre los dos relojes de unos 187 ms. Este valor es más que aceptable para utilizar este sistema en entornos de vigilancia, tal y como se describe en (Karlsson, 1996).

#### 4.4.5.4 Comparación con otros sistemas de compresión

Este último test permitirá evaluar el sistema MIJ2K con respecto a otros códecs de compresión que no utilizan técnicas de compensación de movimiento, y que por tanto son adecuados para los sistemas de vigilancia. Para ello realizaremos una comparación con el códec MJPEG, y la especificación H.264-Intra. Esta última es similar a la especificación H.264 pero no incluiría las técnicas complejas de detección de movimiento, que añadirían grandes retardos en la transmisión.

Se realizarán evaluaciones de calidad usando las métricas PSNR, VQM, y SSIM sobre la secuencia 'Surveillance'. Utilizaremos esta secuencia porque es la más representativa, ya que ha

| Codec | Video Size | Compression Ratio | PSNR | SSIM | VQM |
|---|---|---|---|---|---|
| H.264-Intra | 2567KB | 248.23:1 | 31.7048dB | 0.87921 | 1.66347 |
| J2K | 2556KB | 249.30:1 | 30.9377dB | 0.87876 | 1.72719 |
| MIJ2K | 2534KB | 251.46:1 | 39.5379dB | 0.96434 | 0.79581 |
| MJPEG | 8094KB | 78.72:1 | 28.6561dB | 0.79552 | 3.61294 |

Tabla 4.9: Resultados de la comparación de calidad de compresión de la secuencia 'Surveillance' con los códecs H.264-Intra, J2K, MIJ2K, y MJPEG.

sido capturada directamente por una cámara de videovigilancia. La resolución de esta secuencia es además más representativa con las cámaras de vigilancia actuales.

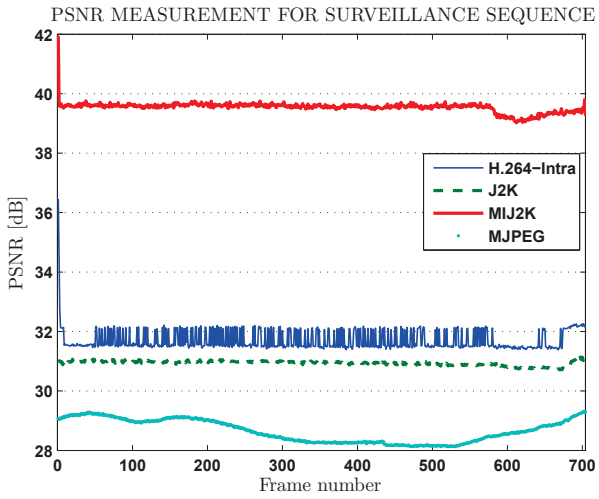


Figura 4.32: Medida de calidad PSNR para los códecs H.264-Intra, J2K, MIJ2K y MJPEG para la secuencia 'Surveillance'.

En las figuras 4.32, 4.33, y 4.34 podemos observar los resultados de la comparación realizada. Podemos ver que el códec MIJ2K proporciona mejores resultados para todas las métricas evaluadas. El resultado de la medida de PSNR el alrededor de 8dBs mejor que el segundo códec de la lista, el H.264-intra. La peor compresión se ha obtenido en este caso por el códec MJPEG, que ha obtenido un PSNR medio de 28dB, cerca de 11dBs menos que el valor obtenido por MIJ2K. Se puede observar además en la tabla 4.9 que la proporción de compresión para el MJPEG es además mucho peor. Principalmente porque no puede alcanzar los niveles de compresión de los otros códecs.

La métrica SSIM también le otorga mejores resultados al códec MIJ2K. Mientras que los resultados proporcionados para los códecs J2K y H.264-intra son más o menos similares. Aunque H.264-intra sigue proporcionando ligeras mejoras. MJPEG sigue siendo el que peor resultado obtiene. En la métrica de comparación subjetiva VQM podemos seguir observando el mismo comportamiento pero a la inversa. En esta métrica, valores mayores indican peor calidad, mientras que los valores más próximos a cero representarían mejores resultados.

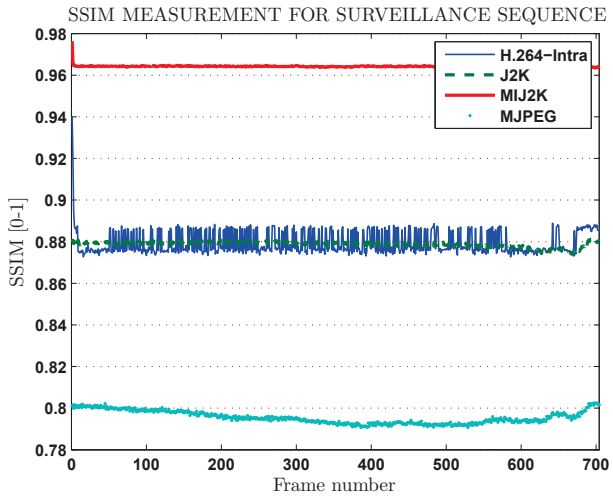


Figura 4.33: Medida de calidad SSIM para los códecs H.264-Intra, J2K, MIJ2K y MJPEG para la secuencia 'Surveillance'.

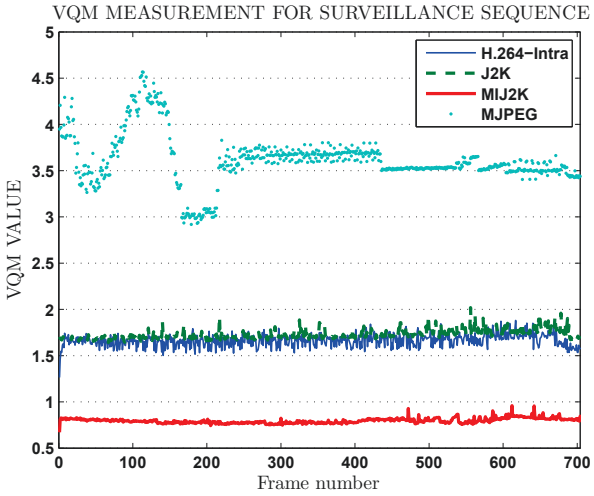


Figura 4.34: Medida de calidad VQM para los códecs H.264-intra, J2K, MIJ2K y MJPEG para la secuencia 'Surveillance'.

Con estos resultados podemos concluir que el sistema de transmisión MIJ2K que hemos desarrollado mejora a los códecs con similares características que no emplean técnicas de compensación de movimiento.

### 4.4.6 Conclusiones

En esta sección hemos descrito al completo una técnica de optimización de compresión y transmisión de secuencias basadas en el método de compresión J2K. Este proceso de optimización surgió al intentar integrar un conjunto de cámaras PTZ en el entorno multi-agente

que estábamos desarrollando. En este entorno contábamos con una serie de servidores y hardware especializado que permitía la compresión por hardware de secuencias J2K. Aprovechamos esta característica y además desarrollamos el proceso MIJ2K, que inclusive hemos conseguido patentar.

En este desarrollo hemos podido comprobar como el proceso de optimización MIJ2K permite introducir un algoritmo de detección de movimiento en tiempo real en un sistema de transmisión basado en J2K. Este es empleado para realizar un envío condicional de diferentes regiones de la imagen utilizando el concepto de cuadrícula que proporciona la compresión J2K. Con esta optimización se puede reducir el ancho de banda necesario para la transmisión, minimizar los tiempos de compresión, y descompresión, así como disminuir la latencia general. Las pruebas realizadas sobre la calidad de la compresión, arrojan grandes diferencias con un sistema J2K tradicional. Permitiendo obtener una calidad mucho superior (sobre 9dB en el mejor caso) para la misma proporción de compresión.

Además este sistema puede integrarse fácilmente con el estándar de transmisión diseñado para el transporte de secuencias J2K descrito en el RFC 5371 (Futemma et al., 2008). Por lo que cualquier cliente compatible podría recibir, descomprimir, y visualizar las secuencias optimizadas por el proceso MIJ2K. Para evaluar este punto hemos desarrollado tanto la parte de servidor como de cliente. Y se ha podido observar la adecuación de este sistema para un entorno de transmisión en tiempo real, como el que requiere un entorno de videovigilancia.

## 4.5 Experimentación

Este tipo de entornos son complicados de evaluar, ya que no contamos con un único algoritmo aislado sobre el que poder medir su resultado. Contamos con un diseño cuyo entorno maneja múltiples componentes distribuidos que tienen que interactuar para conseguir un objetivo específico. Por ejemplo, tenemos procesos de adquisición y procesamiento de imágenes, agentes que monitorizan el entorno y se coordinan, interfaces de usuario que permiten visualizar el comportamiento, etc.

Por lo tanto, hacer un análisis exhaustivo de cada una de las partes por separado no sería lo suficientemente significativo. Por este motivo trataremos de evaluar el comportamiento del sistema multi-agente de manera global, atendiendo al objetivo inicial que planteamos en la descripción del entorno. Esta evaluación lleva asociado un gran coste de desarrollo, ya que será necesario implementar cada una de las partes y hacerlas funcionar en el entorno real.

En esta experimentación llevaremos el diseño del sistema multi-agente que se ha diseñado a nivel teórico, a una implementación práctica sobre el entorno real. Para esto será necesario contar con alguna plataforma intermedia que facilite el desarrollo y despliegue del sistema multi-agente. Estas permiten simplificar la implementación de los agentes, especialmente en las tareas de ejecución distribuida, comunicación, y planificación de tareas.

Para este caso se ha seleccionado la plataforma JADEX (Braubach et al., 2003), que puede ser considerada una extensión de la plataforma JADE (Bellifemine et al., 2001) para los agentes basados en deseos, creencias e intenciones (BDI). El modelo de agentes BDI es probablemente el más conocido y el que mejor permite representar agentes con capacidad de razonamiento (Georgeff et al., 1999). Los agentes que razonan son especialmente proactivos, y permiten tomar la iniciativa en lugar de simplemente reaccionar a los cambios.

De cualquier manera, la elección de una plataforma concreta para implementar el sistema multi-agente dependerá de las condiciones de ejecución y despliegue que nos proporcione el entorno con el que estamos trabajando. Por ejemplo, no es lo mismo contar con un entorno donde es necesario ejecutar los agentes en dispositivos móviles, como contar con ordenadores completos que no cuentan con limitaciones de consumo energético o limitada capacidad de procesamiento. En nuestro caso particular estamos tratando con un entorno multi-cámara donde contamos con múltiples servidores encargados de la gestión autónoma que vamos a desarrollar, por ello podemos utilizar este tipo de plataformas sin problemas.

Tanto la plataforma JADE como JADEX, nos permitirán desplegar agentes de forma distribuida, y además proporcionan un sistema de comunicación entre agentes basado en el estándar FIPA (Foundation for Intelligent Physical Agents). Este se conoce como el lenguaje de comunicación entre agentes (FIPA-ACL). Estas plataformas además cuenta con interfaces gráficas que facilitan el desarrollo y depuración de los agentes, así como la monitorización de

su comunicación.

Utilizar la plataforma JADEX permite que los agentes tengan la habilidad de comunicarse, es decir, puedan transmitirse información. Sin embargo no sólo es necesario poder comunicarse, es necesario que estos puedan entenderse, es decir, hablen el mismo lenguaje. Para solucionar este problema de interacción entre agentes utilizaremos el concepto de ontologías. Una ontología nos permitirán representar el conocimiento del dominio como un conjunto de conceptos en el entorno multi-cámara con el que estamos trabajando.

Por lo tanto, para que los agentes puedan entenderse implementaremos una ontología que básicamente representa el conocimiento que hemos descrito a lo largo de esta aplicación. Para ello utilizaremos la herramienta Protegé, que nos permitirá definir los diferentes conceptos. Además, utilizaremos herramientas adicionales como la OntologyBeanGenerator, que nos permitirán exportar estos conceptos directamente a código de programación que puede ser fácilmente integrado en la plataforma JADEX. Podemos ver un ejemplo del desarrollo de parte de la ontología mediante la herramienta Protegé en la figura 4.35.

Esta herramienta (Horridge et al., 2004) permite la definición de ontologías de una manera sencilla mediante el uso de una interfaz gráfica. Además podemos exportar estas ontologías directamente al ficheros en formato Java, el mismo lenguaje de programación con el que se trabaja en plataformas como JADE o JADEX.

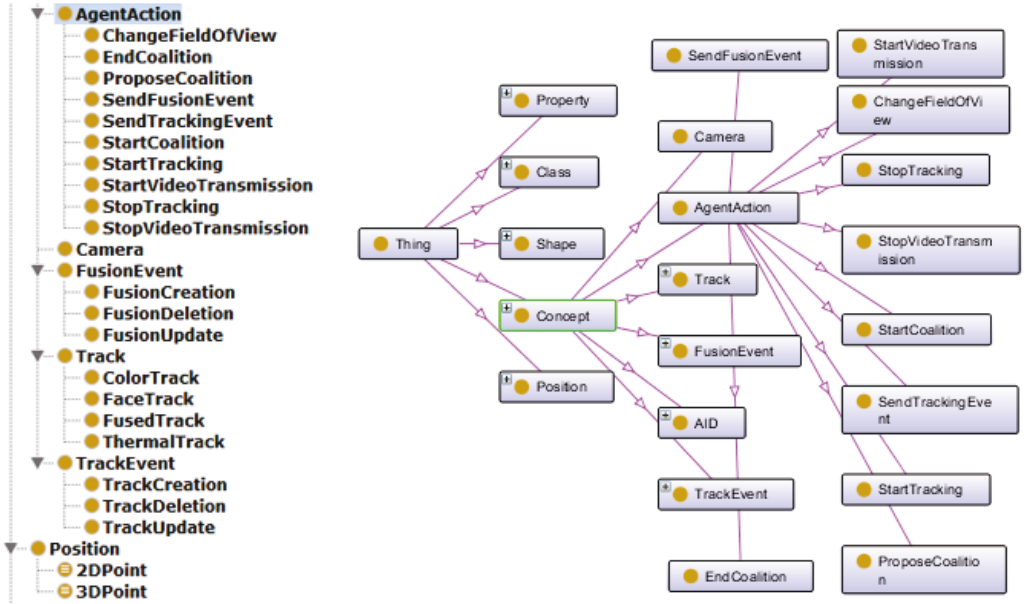


Figura 4.35: Para la aplicación multi-cámara se ha desarrollado una ontología que permitirá a los agentes comunicarse. Para ello se ha utilizado la herramienta Protegé junto con el plugin OntologyBeanGenerator, que permite exportar este conocimiento a Java y por tanto integrarlo en la plataforma JADEX.

Con el uso de JADEX y las ontologías, ya tendríamos solucionados los problemas fundamentales a la hora de implementar el sistema multi-agente en un entorno real. Esto es, tendríamos

la plataforma que nos permite implementar la lógica de gestión de los agentes, su despliegue en el sistema distribuido, y el medio para que los agentes se comuniquen y entiendan.

En el entorno descrito tenemos tres cámaras PTZ, $C_1$, $C_2$, y $C_3$ ubicadas en una posición fija del entorno. Cada una de las cámaras está conectada a un servidor diferente, y en cada servidor se ejecutan los Agentes Sensor y Agentes Control para cada una de las cámaras. Mientras que los Agentes de Interfaz y Fusión se ejecutarán en una cuarta máquina que representa el sistema con el que interactúa el operador. Como podemos ver, contamos con un sistema totalmente distribuido, donde los diferentes agentes se estarían ejecutando en sistemas informáticos diferentes. Todos los sistemas se encontrarían en este caso en la misma red de área local. La relación de estos agentes se describe en la tabla 4.10.

| Agente | Tipo | Entrada | Salida |
|---|---|---|---|
| AgenteCámara1 | Agente Sensor | Control AgenteControl1 | Pistas locales Cámara1, Vídeo |
| AgenteCámara2 | Agente Sensor | Control AgenteControl2 | Pistas locales Cámara2, Vídeo |
| AgenteCámara3 | Agente Sensor | Control AgenteControl3 | Pistas locales Cámara3, Vídeo |
| AgenteControl1 | Agente Control | Pistas Globales AgenteFusion, Coordinación *CA* | Control Cámara1, Coordinación *CA* |
| AgenteControl2 | Agente Control | Pistas Globales AgenteFusion, Coordinación *CA* | Control Cámara2, Coordinación *CA* |
| AgenteControl3 | Agente Control | Pistas Globales AgenteFusion, Coordinación *CA* | Control Cámara3, Coordinación *CA* |
| AgenteFusion | Agente Fusión | Pistas locales: AgenteCámara1, AgenteCámara2, AgenteCámara3 | Pistas Globales |
| AgenteInterfaz | Agente Interfaz | Pistas Globales AgenteFusion, Vídeo | N/A |

Tabla 4.10: Descripción de los agentes instanciados para la experimentación en el entorno de vigilancia multi-cámara.

Para realizar la transmisión de vídeo y el control de las cámaras PTZ, el Agente Sensor utilizará el Controlador de Sensor descrito en 4.3.1.1, que incorpora las optimizaciones de transmisión de vídeo descritas en la sección 4.4. Esta transmisión de vídeo se realiza en tiempo real hacia el Agente Interfaz que obtiene las secuencias de vídeo y las muestra por pantalla para la evaluación por parte del operador. El agente desarrollado para esta interfaz puede observarse

en la figura 4.36, donde se muestran las diferentes fuentes de vídeo generadas por cada una de las cámaras.

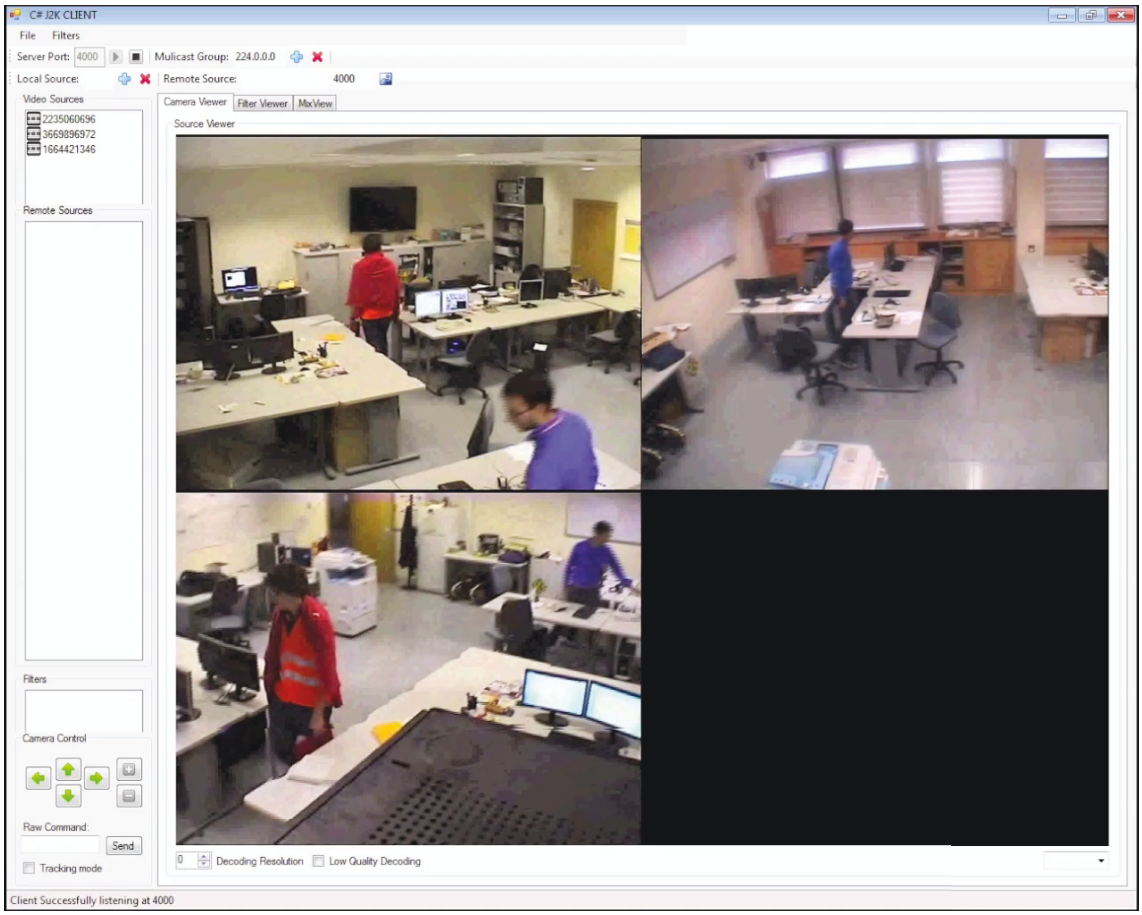


Figura 4.36: Agente Interfaz (*IA*) desarrollado para el entorno multi-cámara, donde se ha incluido la optimización de transmisión de vídeo para poder operar sobre el sistema en tiempo real.

El propósito del prototipo desarrollado mantiene el objetivo inicial propuesto en esta aplicación. En ella, cada una de las cámaras deberá monitorizar las diferentes entidades presentes del objeto. Recordemos que las entidades en este entorno consistían en determinados objetos de color, cuya detección y seguimiento se realizará en cada Controlador de Sensor asociado a cada cámara. Estos controladores se encuentran ubicados en el mismo servidor donde se despliega cada Agente Sensor.

Para evaluar el comportamiento del sistema multi-agente vamos a realizar un experimento donde aparecen diferentes objetivos a ser monitorizados, $T_1$ (Azul), $T_2$ (Rojo), y $T_3$ (Verde). Cada uno de estos objetos tiene un color diferente, lo que le otorgará una prioridad de monitorización diferente. Siendo $P_{ROJO} > P_{AZUL} > P_{VERDE}$. A Continuación se describe la aparición de cada uno de estos objetivos dentro del entorno.

- $T_1$ (Azul): El primer objetivo se encuentra desde el primer momento ($t_1$) dentro del entorno. Como este es el único objetivo detectado del entorno, se convertirá en el objetivo de monitorización de todas las cámaras.

- $T_2$ (Rojo): Este será el segundo objetivo en aparecer en el instante $t_2$. Como este objetivo es visible para todas las cámaras, y es más prioritario que el objetivo azul, estas deberán coordinarse para que dos de ellas monitoricen el objetivo rojo, y otra se quede con el objetivo azul.

- $T_3$ (Verde): Este es el último objetivo que aparecería en el instante $t_3$. Como hay tres objetivos diferentes y tres cámaras, por lo que cada una de las cámaras del entorno deberían encontrarse monitorizando un objetivo diferente.

Este experimento ha sido probado en el entorno real con éxito. Podemos ver algunas capturas de los diferentes instantes de aparición de cada uno de los blancos en la figura 4.37. En estas capturas cada columna representa una cámara en concreto que son $C_1$, $C_2$, y $C_3$ de izquierda a derecha respectivamente. Cada una de las filas representará por tanto cada uno de los instantes donde aparecen los blancos $T_1$, $T_2$, y $T_3$.

En el instante $t_1$, que se observa en la primera fila, podemos ver que todas las cámaras del entorno se encuentran monitorizando al único blanco detectado $T_1$. En el instante $t_2$, que es cuando aparece el blanco $T_2$, debido a las prioridades de los colores, podemos ver que dos de las cámaras han comenzando a monitorizar a $T_2$, mientras que la cámara central ha mantenido el blanco $T_1$ (si observamos la segunda fila). Finalmente, en el instante $t_3$, aparecerá el blanco $T_3$. Como contamos con tres blancos para tres cámaras, cada cámara debería encontrarse monitorizando un blanco diferente, tal y como podemos ver en la tercera fila. Se puede apreciar que la cámara $C_1$ está monitorizando $T_2$ (rojo), la cámara $C_2$ ha mantenido el blanco $T_1$ (azul), mientras que la cámara $C_3$ se ha quedado con el blanco $T_3$ (verde).

Para comprender mejor el experimento, así como permitir observar la monitorización realizada por los diferentes agentes, contamos con una grabación en vídeo que demuestra el funcionamiento del sistema en tiempo real. Este vídeo está disponible en el o popular servicio de transmisión Youtube[2].

Este experimento, que a priori puede parecer sencillo, ha supuesto una compleja tarea de desarrollo e integración del sistema multi-agente, así como de las diferentes partes que la componen. Además, la idea de este experimento no es evaluar exhaustivamente el diseño concreto que proponemos a modo de ejemplo para la aplicación multi-cámara. La tarea de evaluación consistía en probar que el sistema diseñado se puede implementar en un entorno real, y que los agentes permiten realizar una gestión en tiempo real de las cámaras del entorno.

[2] https://www.youtube.com/watch?v=S4r2JTstp_k

En este sentido, el resultado de la experimentación ha sido satisfactorio, y nos ha permitido comprobar que el diseño del sistema multi-agente desarrollado se puede integrar en un entorno real de este tipo. Si bien es cierto que su desarrollo e integración ha resultado ser bastante compleja, la mayoría del trabajo desarrollado podría reutilizarse para su aplicación en diferentes entornos con diferentes cámaras, o mecanismos de coordinación.

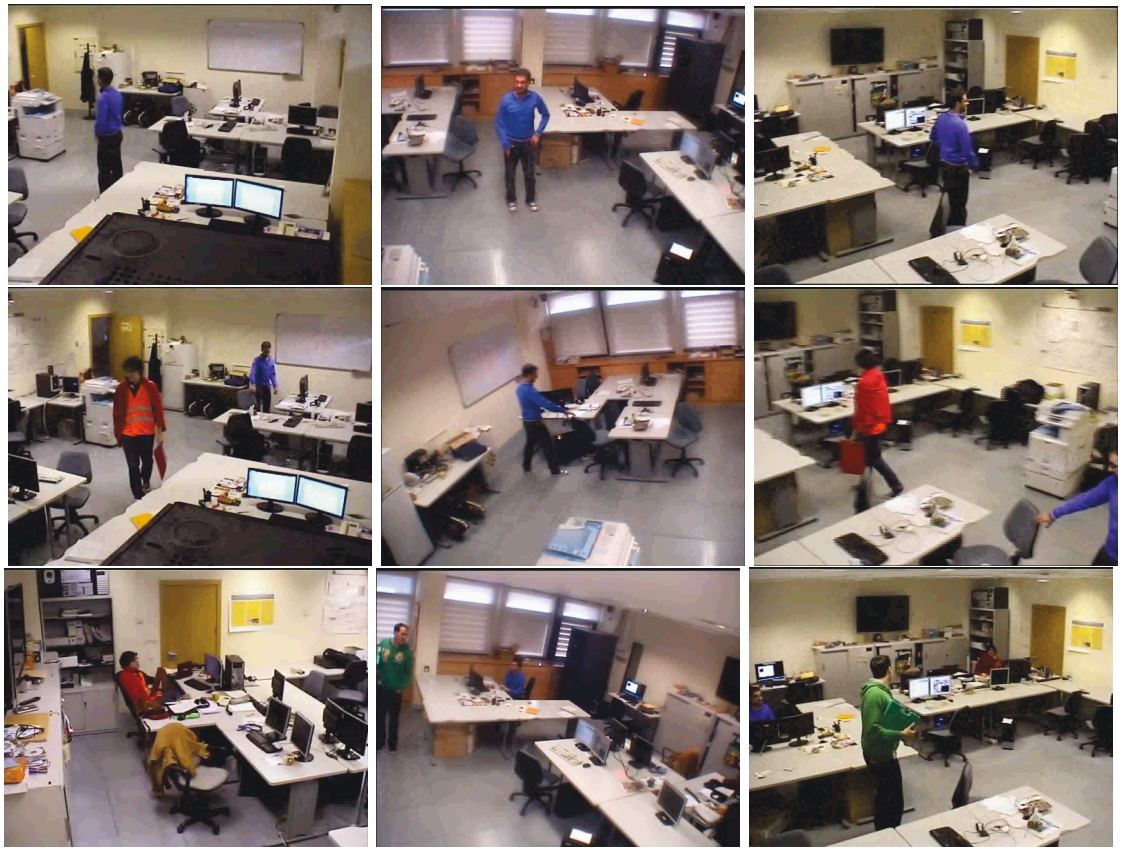

Figura 4.37: Resultado de la experimentación del sistema multi-agente en el entorno multi-cámara según aparecen nuevos objetivos en el entorno. Cada una de las columnas representa el campo de visión de cada una de las cámaras, $C_1$, $C_2$, y $C_3$, de izquierda a derecha. Cada fila representa un instante de tiempo, $t_1$, $t_2$, and $t_3$ de arriba a abajo. En cada figura podemos observar el objetivo de monitorización de cada cámara para cada instante de tiempo.

## 4.6 Conclusiones

En esta aplicación se ha trabajado sobre un entorno real de videovigilancia que cuenta con un conjunto de cámaras PTZ distribuidas en diferentes sistemas informáticos. Este tipo de entornos pueden llegar a suponer un desafío, sobre todo cuando pasamos de la teoría a la práctica. En esta hemos tratado con la gestión autónoma de las cámaras para la monitorización automática de determinadas entidades del entorno.

Para ello hemos utilizado el diseño del sistema multi-agente general propuesto en la sección 3, que ha sido adaptado para su aplicación al entorno multi-cámara descrito. La adaptación ha consistido en seleccionar los diferentes agentes necesarios para realizar los objetivos propuestos, y diseñar e implementar cada una de las particularidades para este entorno.

Durante el desarrollo de esta aplicación hemos tenido que trabajar a diferentes niveles. Desde los procesos de más bajo nivel, como puede ser la interacción con el dispositivo físico de las cámaras, a las tareas de más alto nivel que representan la coordinación mediante el sistema multi-agente. Trabajar con un entorno real nos ha permitido evaluar nuestro diseño en un entorno de videovigilancia real, por lo que el diseño no se queda en algo puramente teórico. Por ejemplo, la integración de las cámaras, tal y como hemos visto en la sección 4.3.1.1, y 4.4, ha supuesto un gran desafío, pero ha permitido obtener nuevas líneas de investigación, que inclusive han resultado en la publicación de una patente.

Para evaluar el entorno multi-cámara diseñado hemos implementado el sistema multi-agente que se integra en el entorno real, que es la mejor manera de evaluar su correcto funcionamiento. En el experimento de coordinación se ha podido observar el correcto funcionamiento de los diferentes procesos diseñados, así como la coordinación entre los agentes para monitorizar las entidades del entorno. Conseguir este resultado en un entorno real supone una valiosa prueba de la adecuación del diseño realizado, incluso para un entorno tan exigente como el de la videovigilancia.

# 5

# Entorno Marítimo

## 5.1 Introducción

Los sistemas de vigilancia marítima, normalmente gestionados por el ejército o la policía, se utilizan diariamente para identificar e interceptar amenazas en puertos, zonas costeras, fronteras, plataformas marítimas, o cualquier otra instalación de importancia. La monitorización de este tipo de entornos, que por lo general suelen ser muy extensos, requieren de un gran despliegue de sensores encargados de la detección de los objetos de interés del entorno. En este tipo de entornos podemos encontrarnos con diferentes tipos de sensores, tanto colaborativos como no colaborativos. Dentro de los sensores colaborativos contamos con los Sistemas de Identificación Automática (AIS), en los que cada embarcación es la responsable de enviar su posición, junto con otros parámetros como la velocidad y el rumbo, a una estación que recoge y monitoriza las posiciones de los barcos. Los sensores no colaborativos, serían aquellos que no requieren de la buena voluntad de los patrones para ser detectados. El sensor más habitual para esta tarea son los sensores radar que se suelen ubicar en las costas o puertos marítimos. También es normal encontrar sistemas de vigilancia con cámaras (CCTV) para complementar de forma visual el estado del entorno. Toda esta información es normalmente integrada en uno o varios puestos donde se monitorizan y evalúan en tiempo real las posibles amenazas. Estos sistemas son llamados Servicios de Trafico Marítimo (del inglés Vessel Traffic Service, VTS).

Los sistemas AIS son sensores integrados en cada embarcación que periódicamente envían información del barco, como su identidad, posición actual, rumbo, y otra serie de parámetros de interés. Todos ellos cuentan con algún tipo de sistema de localización por satélite, como el GPS, Glonass, o Galileo. Por el contrario, los sistemas radar son sensores no colaborativos que monitorizan el entorno periódicamente utilizando pulsos de energía que son reflejados por los diferentes elementos presentes en el entorno como las embarcaciones. Estos sensores se encuentran instalados en posiciones estratégicas, dependiendo principalmente de la orografía concreta del terreno. Es habitual poder encontrar múltiples sensores radar cubriendo las mismas zonas del entorno, ya que esto mejora la posibilidad de detección y en algunos casos la precisión

de la posición cuando se usan sistemas de fusión de información. Este es un problema muy estudiado por la comunidad de fusión multi-sensor (Mitchell, 2007), y la mayoría de sistemas de vigilancia marítimos actuales incorporan sistemas de fusión de información para facilitar el análisis y la toma de decisiones en entornos marítimos (García et al., 2010).

Algunos sistemas también incluyen sensores de visión de largo alcance, normalmente cuando la seguridad es un elemento de vital importancia. Ya no sólo interesa conocer la ubicación de las embarcaciones para detectar posibles peligros, si no que también interesa poder visualizar su actividad. Estos requisitos se han incrementado en los últimos años debido a la creciente actividad de la piratería marítima (Murphy, 2013), donde continuamente se ponen en peligro embarcaciones, bienes, tripulación, pasajeros, etc. Para ello se suele contar con cámaras PTZ de largo alcance controladas por el operador, encargado de seleccionar y monitorizar los posibles elementos de interés. Algunas cámaras más avanzadas pueden integrar sensores externos, como radares, que permiten dirigir automáticamente la cámara hacia las embarcaciones que están siendo detectadas por el sensor (Systems, 2009). Aunque estos sistemas son sofisticados, tienen grandes limitaciones cuando pensamos en entornos complejos con múltiples radares, estaciones AIS, y cámaras PTZ. Aquí la tarea de gestión de las cámaras se complica debido a la gran cantidad de información percibida del entorno, las diferentes entidades a monitorizar, la posible coordinación que podrían realizar las cámaras para no seguir a las mismas embarcaciones, etc.

Este tipo de entornos por tanto supone un gran desafío a varios niveles. El primero a nivel de integración, ya que contamos con múltiples sensores heterogéneos distribuidos espacialmente por el entorno. El segundo a nivel de integración de información, ya que contaremos con información de múltiples sensores que se encuentran monitorizando las embarcaciones del entorno. Y el tercero a nivel de gestión y coordinación de las cámaras PTZ, que deberían ser capaces de explotar la información proporcionada por el entorno para realizar una gestión eficiente en la monitorización de las diferentes embarcaciones.

Para ello en esta aplicación concreta se pretende diseñar un sistema de control automático de cámaras PTZ en función de las prioridades del operador del sistema VTS. La idea es poder aplicar el sistema multi-agente diseñado en esta tesis e integrarlo en un sistema con múltiples tipos de sensores, a diferencia de la aplicación del entorno multi-cámara. De esta forma podemos evaluar el diseño multi-agente en un entorno que cuenta con diferentes características tanto a nivel de sensores, como a nivel de fusión de información. El control de las cámaras sin embargo será muy similar al descrito en el entorno multi-cámara. Y en este caso, basarnos en un diseño general nos permitirá reutilizar gran parte del trabajo de gestión y coordinación realizado para el entorno multi-cámara.

Podemos encontrar trabajos similares que se centran en la gestión de cámaras PTZ en función de las percepciones percibidas por el entorno, como los descritos en (Bustamante et al., 2013), (Luis Bustamante et al., 2014b), (Doyle et al., 2013), y (Singh et al., 2012). Sin embargo ninguno de esos trabajos contempla el uso de sensores externos para mejorar la

gestión y la coordinación de las cámaras PTZ. Como en el entorno marítimo contamos con múltiples fuentes de información, tiene sentido poder utilizarlas para ayudar en la tarea de gestión. En esta aplicación se evaluará cómo el uso de sensores externos, así como los sistemas de fusión de información, permiten mejorar la tarea de vigilancia frente a sistemas únicamente basados en la información generada por las cámaras.

La evaluación del sistema multi-agente en un entorno marítimo real, como en el que se basó el entorno multi-cámara, resulta una tarea mucho más complicada. Principalmente porque estos sistemas tan caros y costosos de instalar son propiedad de empresas o entidades gubernamentales. Esto supone que el acceso para experimentación y grabación de datos se encuentra muy restringido, ya que son sistemas que se encuentran en funcionamiento y pueden contener información sensible. Durante el desarrollo de la tesis hemos podido trabajar con entornos de vigilancia marítimos reales A.1 y hemos podido constatar la complejidad de estos sistemas, y la poca disponibilidad por parte de las empresas en suministrar información, así como de integrar sistemas experimentales. En este sentido, durante el desarrollo de estos proyectos se diseñó una herramienta de simulación de trayectorias de embarcaciones con el fin de probar la lógica de los algoritmos de fusión ante diferentes circunstancias. Esta herramienta se extendió de forma notable para el desarrollo de esta tesis, y actualmente cuenta con simulador de radares, estaciones AIS y cámaras. También se integraron los agentes descritos en el diseño multi-agente, lo que nos permitió evaluar mejor el comportamiento del sistema multi-agente. Esta herramienta se describe con más detalle en la sección 5.4.

El resto de la sección se organiza de la siguiente forma. En la sección 5.2 se describe en más detalle el entorno con el que vamos a trabajar, así como los objetivos específicos de la aplicación. En el punto 5.3 podemos encontrar la aplicación del sistema multi-agente definido en 3 para el entorno concreto de la vigilancia marítima. El diseño del simulador de entornos marítimos queda definido en la sección 5.4, así como la experimentación llevada da cabo en la sección 5.5.

## 5.2 Descripción del entorno y objetivos

Como se ha descrito en la introducción, el diseño de esta aplicación está centrado en los entornos de vigilancia marítima. En concreto en la aplicación de un sistema multi-agente para el control de un conjunto de cámaras PTZ distribuidas por el entorno. Dependiendo de la información percibida por los sensores del entorno (radar y AIS), el sistema multi-agente deberá gestionar y controlar las cámaras PTZ. Esta gestión consistirá en el seguimiento automático de las embarcaciones presentes en el entorno de acuerdo a una serie de prioridades que pueden estar ajustadas por el operador. Esto permitirá reducir la carga de trabajo a los operadores cuando necesiten gestionar y coordinar un conjunto de cámaras PTZ.

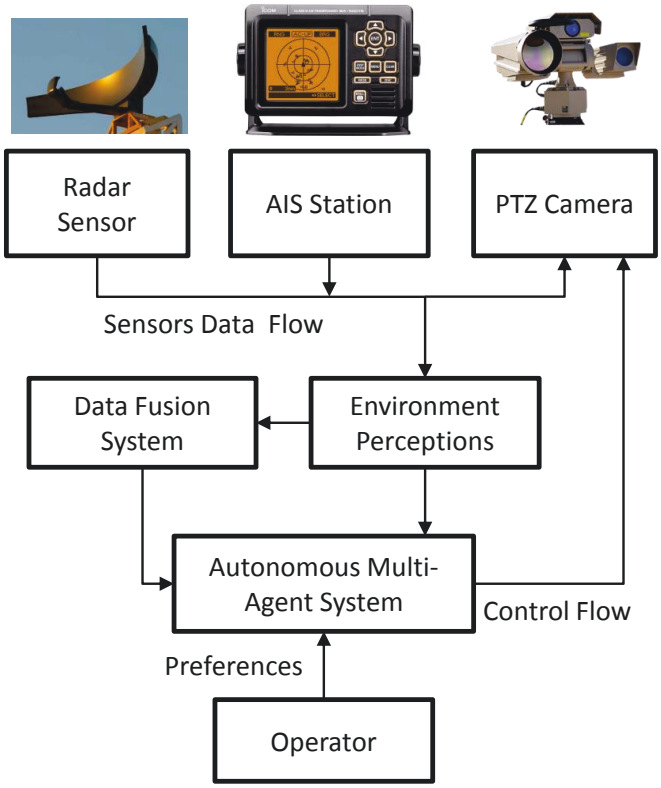


Figura 5.1: Descripción general del entorno marítimo y su relación con el sistema de fusión. Podemos encontrar sensores como radares, estaciones AIS, y cámaras PTZ. Estas últimas serán gestionadas y coordinadas con la aplicación del sistema multi-agente.

Este entorno se define brevemente en la figura 5.1. En ella podemos observar la disposición de diferente sensores del entorno marítimo como el radar, el AIS, y las cámaras PTZ. Sobre este conjunto de sensores hay un sistema multi-agente que recibe las percepciones del entorno y envía señales de control a las cámaras PTZ. La fusión de datos también se encontraría recibiendo información de los diferentes sensores, y se puede considerar como una fuente de datos adicional que podría recibir el sistema multi-agente. Este se encargaría de analizar las

percepciones, razonar acerca del estado del entorno y las prioridades del operador, y finalmente aplicar comandos de control en las cámaras para monitorizar los diferentes embarcaciones.

El objetivo final del sistema multi-agente es coordinarse en la monitorización de las embarcaciones, de tal forma que cada una de las cámaras se encuentre monitorizando un barco diferente. Para ello, los agentes encargados del control de las cámaras necesitarán tomar la información generada por el sistema de fusión, seleccionar una entidad a monitorizar, y coordinarse con el resto de agentes en caso de presentarse un conflicto. Si no hubiese embarcaciones a monitorizar, o las embarcaciones se encuentran fuera de su rango de cobertura, las cámaras podrán moverse automáticamente por las zonas de monitorización especificadas por el usuario. Este objetivo podría representar un modo de funcionamiento real, donde las cámaras son automáticamente controladas para monitorizar las embarcaciones de más prioridad desde el punto de vista del operador.

## 5.3 Diseño del sistema multi-agente

En esta sección se describe la integración de la arquitectura propuesta en la sección 3 para el entorno de vigilancia marítima descrito. Para ello definiremos la adaptación de cada agente general del diseño, a los agentes más específicos que representan el entorno con el que estamos trabajando.

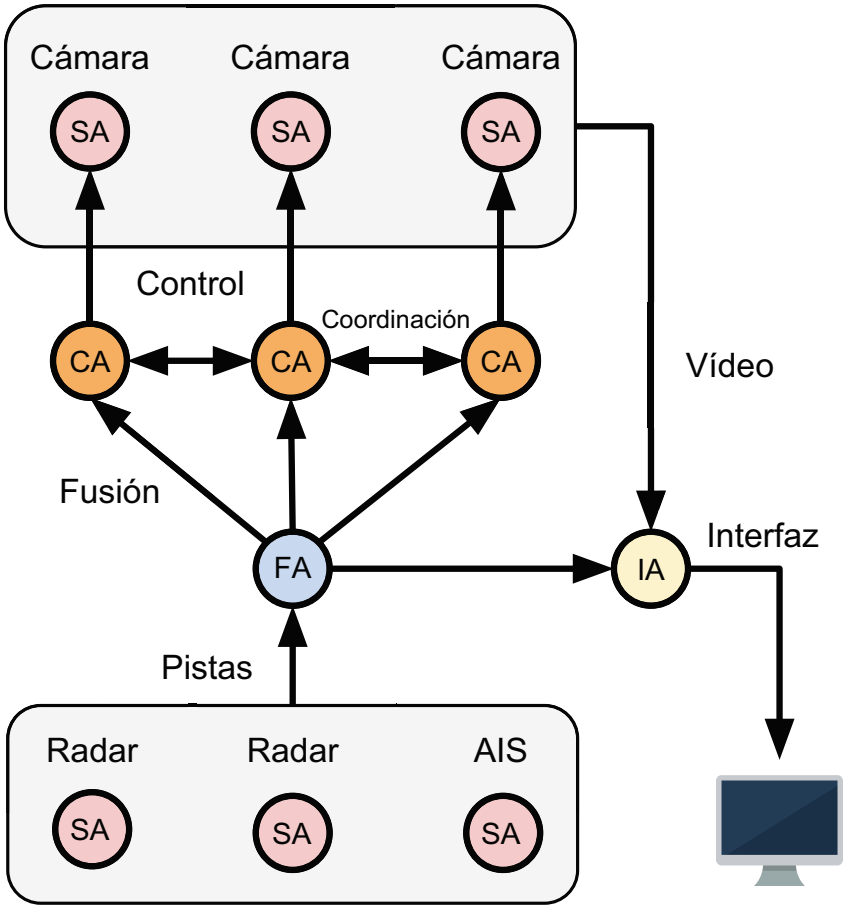


Figura 5.2: Despliegue de arquitectura multi-agente para el control de tres cámaras PTZ utilizando información fusionada de sensores radar y AIS. Podemos encontrar tres Agentes Sensor (*SA*) representando las cámaras. Un Agente Fusión (*FA*) que combina la información de los *SA* que representan al radar y al AIS. Por último podemos encontrar tres Agentes de Control (*CA*) que controlan las cámaras PTZ y se coordinan en función de la información recibida a través del *FA*.

Podemos considerar un conjunto de agentes como los mostrados en la figura 5.2. Al tener que gestionar un conjunto de cámaras PTZ, como en el caso del entorno multi-cámara, contaremos con un conjunto de Agentes Sensor (*SA*) que serán los responsables de representar a los sensores de visión dentro de la arquitectura. Estos agentes, en esta aplicación en concreto, se limitarían únicamente a proporcionar las imágenes del entorno al operador, aunque se experimentará con las cámaras como si fueran una fuente de información más. El entorno será en este caso

monitorizado por los sensores radar y AIS, por lo que habrá más *SA* que los representen e integren su información en la arquitectura multi-agente. Debido a la naturaleza de este tipo de entornos es necesario fusionar la información que estos proporcionan, y para ello se utilizará un Agente Fusión (*FA*).

A diferencia del entorno multi-cámara donde se realizaba una fusión únicamente por los atributos de color de las pistas, en este caso trataríamos con un *FA* especializado en la fusión de posiciones reportadas por los diversos sensores. Esta información fusionada será transmitida a los Agentes de Control (*CA*) que ya sí se encargarán de controlar las cámaras para seguir a las diferentes embarcaciones del entorno. Estos necesitarán coordinarse para realizar a cabo la tarea de forma óptima.

En las siguientes secciones se describe en más detalle los diferentes agentes involucrados para esta aplicación en concreto.

### 5.3.1 Agente Sensor

A diferencia del entorno multi-cámara descrito en el capítulo 4, en este entorno contamos con tres tipos diferentes de sensor. Los sensores radar, las estaciones AIS, y las cámaras PTZ. De cara al diseño, esto se traduce que tendremos que instanciar tres tipos diferentes de Agente Sensor, uno por cada tipo de sensor. El *SA* para las cámaras será muy similar al descrito en el entorno multi-cámara, mientras que los agentes para los sensores radar y AIS, varían ligeramente, teniendo en cuenta que sobre estos no vamos a aplicar ningún tipo de gestión. La descripción de estos agentes se pueden observar en la figura 5.3.

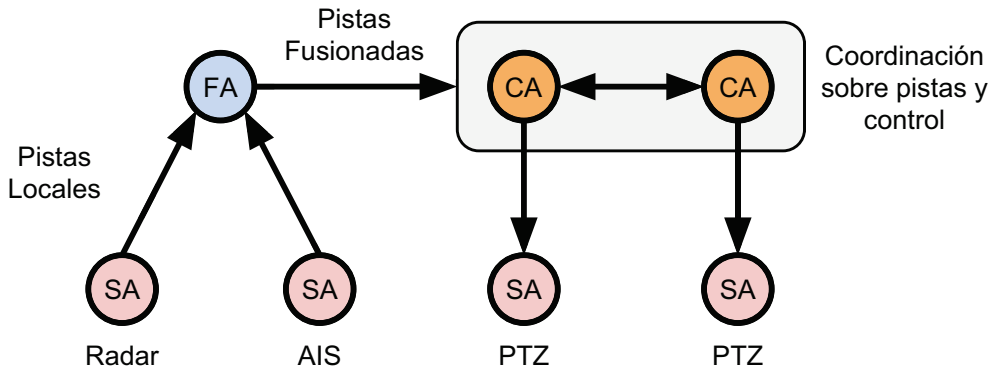


Figura 5.3: En el entorno marítimo contaremos con tres tipos diferentes de Agente Sensor (*SA*), uno por cada tipo de sensor: Radar, AIS, y cámaras PTZ. Cada uno especializado en el acceso y la gestión del sensor que representan.

Los agentes para el radar y el AIS, serán agentes que se encarguen únicamente de capturar la información que estos son capaces de percibir, y transmitirla hacia el resto de agentes de la arquitectura, como el Agente Fusión. Las cámaras PTZ, representadas por otro agente diferente,

recibirán el control por parte de los Agentes de Control que se encuentran razonando sobre la información percibida por el entorno a través del Agente Fusión. De este modo contamos con tres Agentes Sensor diferentes, que se describen a continuación.

#### 5.3.1.1 Agente Sensor AIS

Este agente será el encargado de representar a una estación AIS instalada en el entorno marítimo. Se encargará de recibir los mensajes generados por las diferentes embarcaciones y suministrar esta información al resto de agentes de la arquitectura. Se puede considerar como un agente que permite integrar la información proporcionada por una estación AIS dentro del entorno multi-agente.

Este agente, para esta aplicación en concreto, no recibirá control de ningún otro agente, ya que por el momento no estamos contemplando realizar ajuste alguno sobre la información proporcionada por las estaciones AIS. Sin embargo podríamos llegar a pensar en la aplicación de máscaras de filtrado por zonas, por el identificador de la embarcación, su dimensión, nombre, etc. Esto podría tener utilidad en entornos muy saturados donde nos interese centrarnos en determinadas regiones o características. Dependiendo del entorno es posible llegar a ver casi un millar de blancos con una sola estación AIS.

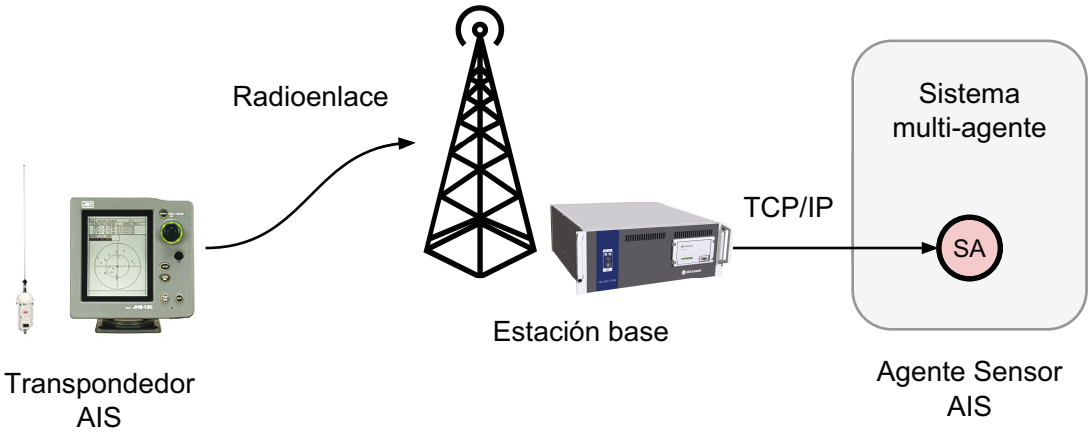


Figura 5.4: Representación del despliegue del Agente Sensor AIS. Este se encontraría conectado a una estación base por una conexión TCP/IP, que se encargaría de transmitirle la información de posición reportada por cada uno de los transpondedores instalados en las embarcaciones.

En un entorno real, similar al que se puede observar en la figura 5.4, este agente se encontraría conectado a la estación base a través de un protocolo de comunicación basado en TCP/IP, por el que recibiría los mensajes que generan las embarcaciones sobre de su posición. La mayoría de estaciones base del mercado actualmente soportan interfaces de red o de puerto serie por las que transmiten la información recibida en tiempo real. Los mensajes se generan en

las propias embarcaciones de forma periódica y son enviados a través a de un radioenlace en la banda VHF. La estación base decodifica la señal y reconstruye y valida la información del mensaje (se aplican algoritmos de comprobación de redundancia cíclica). Los mensajes suelen contener información útil sobre la embarcación, como su posición actual, la velocidad, el rumbo, el puerto de destino, el identificador, etc. Para el objetivo de nuestra aplicación en concreto nos interesará principalmente la información de posición y velocidad. Definimos, por tanto, la información de entrada y salida de este Agente Sensor en la tabla 5.1.

| **Entrada** | |
|---|---|
| **Petición de información** | Generada por cualquier agente de la arquitectura que necesite información de las pistas AIS detectadas en el entorno. |
| **Mensajes AIS** | Mensajes AIS percibidos por la estación base. |
| **Salida** | |
| **Pistas locales AIS** | Conjunto de pistas detectadas por la estación AIS a la que está representando el agente. Esta información contará con información como la posición del barco, su velocidad, su dimensión, la precisión de la medida (si cuenta con sistema de posicionamiento diferencial), etc. |

Tabla 5.1: Información de entrada y salida del Agente Sensor AIS (*SA*) para el entorno marítimo.

#### 5.3.1.2 Agente Sensor Radar

Este agente será muy similar al agente Sensor AIS, salvo que en este caso la información se recogerá de un sensor radar. Los sistemas de radar actuales permiten procesar el vídeo radar y generar un conjunto de objetos detectados en el entorno. De forma similar a la que puede hacerlo un sistema AIS. Con la salvedad de que las pistas no vienen identificadas como tal, si no que normalmente se describen con un identificador numérico local al sensor que está realizando la detección. Al igual que una estación AIS, los radares modernos cuentan con interfaces de red que permiten acceder a la información generada por el radar en tiempo real. Esta información es capturada por el agente e introducida del mismo modo dentro de la arquitectura multi-agente.

Tampoco contemplaremos la gestión o el control de este tipo de sensores, aunque en este caso este sensor en particular tiene mucho mayor margen de ajuste que una estación AIS. Sería posible pensar en un Agente de Control que pudiese estar recibiendo información meteorológica del entorno y analizando la salida generada por el radar, para realizar un ajuste fino sobre sus parámetros. El funcionamiento de un sensor radar depende completamente de las condiciones meteorológicas del entorno, ya que no un radar no se comportará igual con lluvia,

niebla, e incluso olas. En estos casos es necesario ajustar determinados perfiles que modifiquen parámetros como la velocidad de rotación de la antena, el ancho del pulso, etc. Para estos casos de uso, el Agente Sensor Radar debería proporcionar las interfaces de control adecuadas para permitir el ajuste de los parámetros.

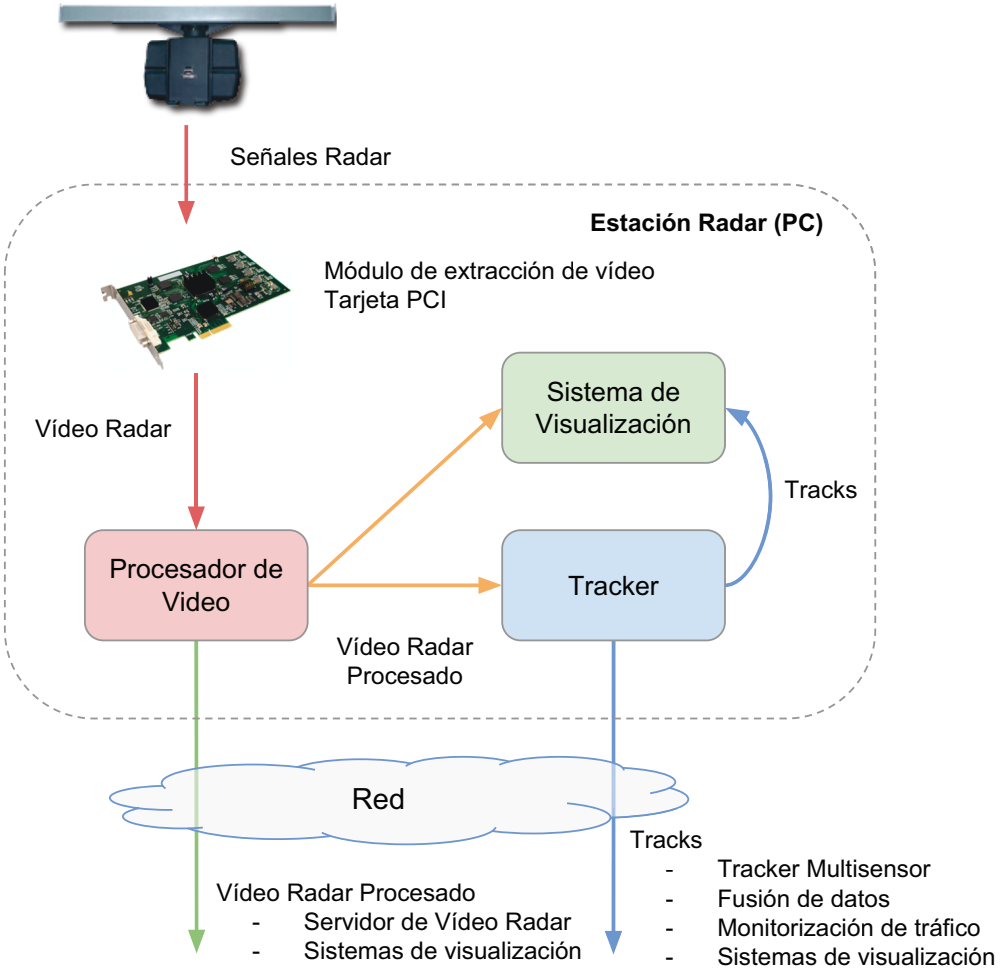


Figura 5.5: Representación de un sistema Radar y su integración en la arquitectura multi-agente a través del Agente Sensor Radar. Este agente se encargaría de recoger la información del radar ya procesada por su extractor, permitiendo acceso a los datos al resto de agentes de la arquitectura.

Este agente debería especializarse para trabajar con un tipo de radar en particular, sobre todo a la hora de la integración física con el sensor, el conocimiento de los protocolos de datos, la representación de la información, etc. Se podría pensar en este entorno también en el diseño de una arquitectura intermedia similar a la desarrollada en el entorno real del sistema multi-cámara con el Controlador de Sensor descrito en la sección 4.3.1.1. Sin embargo no contamos con radares reales que podamos integrar y sobre los que probar un diseño general.

El funcionamiento de un sistema radar difiere por completo al funcionamiento de un sistema

como el AIS. Estos son sistemas más complejos y requieren de cierta lógica de procesamiento para extraer la información del entorno. Normalmente sobre la señal de vídeo generada por el radar se aplican múltiples algoritmos de detección y seguimiento de pistas. Estos algoritmos y cadenas de procesamiento suelen estar suministradas por los mismos fabricantes que suministran el equipamiento hardware. Por tanto, el Agente Sensor para un sensor radar no se encargaría de procesar la información en bruto proporcionada por el radar, si no que utilizaría la información refinada que es generada por los algoritmos de seguimiento con los que pudiera disponer el extractor.

Este enfoque sería similar al Controlador de Sensor diseñado para el entorno multi-cámara, ya que este se encargaba de procesar las secuencias de vídeo y generar un conjunto de pistas. Estos sistemas permiten extraer la información procesada a través de diversas interfaces, como interfaces de red 5.5, y sobre estas actuará el Agente Sensor, capturando y retransmitiendo la información más relevante para el resto de agentes.

De forma general, podemos especificar la información de entrada y salida básica necesaria para integrar el radar la arquitectura multi-agente según se refleja en la tabla 5.2.

| | |
|---|---|
| **Entrada** | |
| **Petición de información** | Generada por cualquier agente de la arquitectura que necesite información de las pistas Radar detectadas en el entorno. |
| **Pistas Radar** | Generadas por el extractor del radar. Contendrá información sobre las entidades detectadas, sus posiciones, dimensiones, estado de la pista, precisión (calculada en función de la dimensión de la celda de resolución donde se encuentre la embarcación), etc. |
| **Salida** | |
| **Pistas locales Radar** | Conjunto de pistas detectadas por el sensor Radar que está representando el agente. Esta información contará con información básica como la posición del barco, velocidad, dimensión, y precisión de la medida. |

Tabla 5.2: Información de entrada y salida del Agente Sensor Radar (*SA*) para el entorno marítimo.

#### 5.3.1.3 Agente Cámara PTZ

Este agente es igual en diseño al Agente Sensor propuesto en el capítulo 4. Al basarnos en el mismo diseño general del sistema multi-agente podemos reutilizar gran cantidad de los

componentes diseñados. En este caso podríamos integrar el agente como un elemento más de la arquitectura.

### 5.3.2 Agente Fusión

El Agente Fusión para el entorno marítimo representa al sistema de fusión de datos encargado de integrar la información de posición a nivel 1 del modelo JDL. Este agente recibirá las detecciones generadas por los diferentes sensores y generará una salida de información fusionada. La diferencia principal con el Agente de Fusión de la aplicación 4 es que en este caso contamos con más variedad de información a fusionar, como la posición del blanco, el error en la medida, y algunas características adicionales como la dimensión. Además, esta información puede se proporcionada por sensores de diferente naturaleza. A modo des esquema este agente este se relacionaría en la arquitectura tal y como se describe en la figura 5.6.

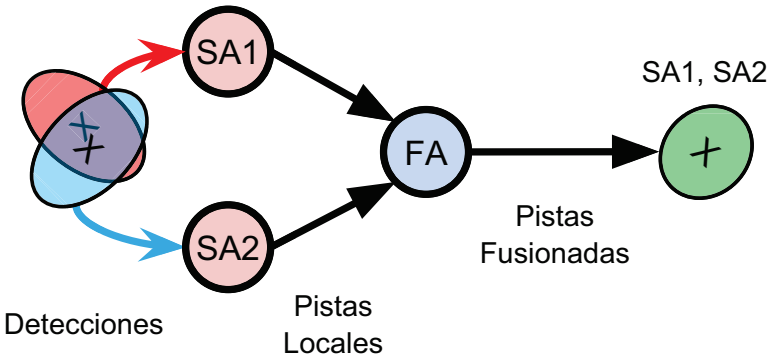


Figura 5.6: Agente Fusión (*FA*) que se encontraría combinando las medidas de posición suministradas por los diferentes sensores como el radar y el AIS. La salida del *FA* consistiría en un conjunto de pistas no redundantes.

El diseño completo del sistema de fusión de datos para el entorno marítimo se escapa del objetivo de esta tesis. Sin embargo, durante el desarrollo de la misma hemos podido trabajar en proyectos con empresas donde hemos diseñado sistemas de fusión completos para entornos de vigilancia marítima. En concreto hemos desarrollado un sistema inicial que publicamos en (García et al., 2010), y otro más reciente y más configurable que se también publicamos en el artículo (Marti et al., 2014). Estos proyectos que se describen brevemente en el anexo A.1, aparte de servir de inspiración para la selección de esta aplicación, han servido para entender profundamente el uso de los sistemas de fusión, así como la algoritmia subyacente. A modo ilustrativo se proporcionará información básica del diseño general de este tipo de sistemas, y cómo se realizaría su adaptación al entorno multi-agente.

En la mayoría de estos diseños se ha optado por un procesado de los sensores de forma distribuida. Es decir, cada sensor es responsable de procesar y generar un conjunto de información ya refinado sobre las pistas detectadas. Esto es, proporcionar información de nivel 1 del modelo

JDL. Para ello necesitan aplicar diferentes algoritmos de enventanado, asociación y filtrado que permitan generar un conjunto de pistas locales. Ya hemos visto que esto podría ser realizado por los sistemas radar comerciales de forma automática.

Estas pistas se enviarían a un centro de fusión de información que de forma centralizada integra la información proporcionada por todos los sensores. Este esquema puede observarse en la figura 5.7, y es el que hemos utilizados en los diferentes proyectos de fusión en entornos marítimos. Hasta ahora este proceso se correspondería con los Agentes Sensor que son los que se encontrarían accediendo a la información del sensor y proporcionándosela al Agente Fusión.

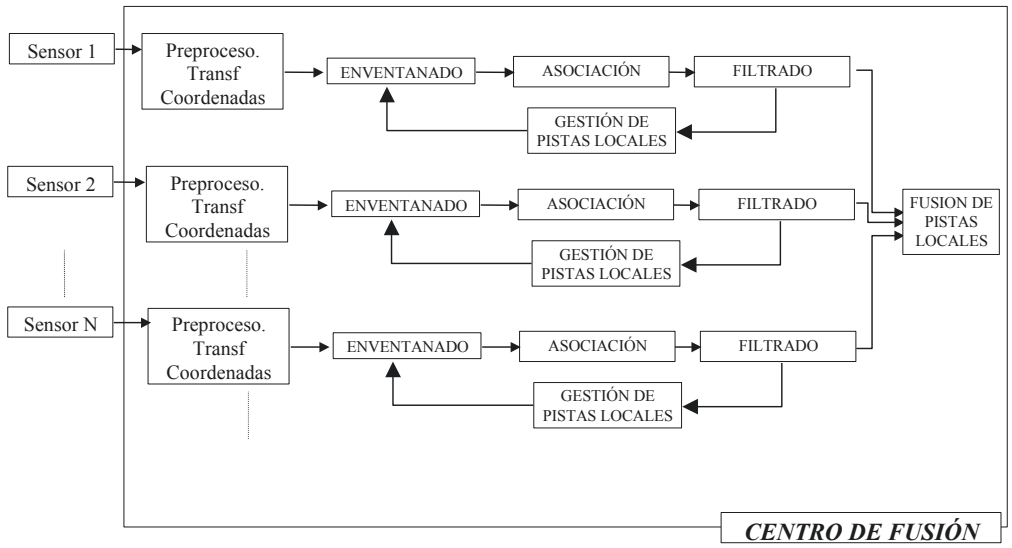


Figura 5.7: Esquema de procesado de información de los sensores de forma distribuida. Cada Sensor se procesa de forma independiente, y el resultado es transmitido a un centro de fusión que se encarga de integrar la información. Este centro de fusión se representa mediante el Agente Fusión, mientras que los sensores y su cadena de procesado mediante los Agentes Sensor.

El Agente Fusión por tanto tomaría la información generada por los Agentes Sensor, y aplicaría un algoritmo de fusión centralizado que permite gestionar las pistas detectadas del entorno. A grandes rasgos se pueden describir algunas de las tareas a realizar por el Agente Fusión en la figura 5.8. Aquí podemos observar cómo el proceso de fusión comienza procesando la información suministrada por los diferentes sensores. El primer paso incumbe ejecutar algunas tareas de mantenimiento de pistas, como la comprobación de pistas globales redundantes que puedan ser recombinadas, la comprobación de integridad de las pistas fusionadas que pueda suponer una ruptura, o incluso el borrado de las pistas globales que se hayan quedado sin componentes locales.

En el segundo paso se coge el conjunto de pistas locales que se han actualizado en el último ciclo de fusión. Si una pista ya se encontraba asociada a una pista global, esta contribuirá a su actualización. Si la pista local de un sensor es una nueva aparición, pasa por un proceso de enventanado y asociación para intentar fusionarlas con las pistas globales existentes. Si no ha

sido posible su asociación, se creará una pista global nueva cuya única componente sea la pista local. Si se ha podido asociar, se incorporará la pista local a la pista global existente.

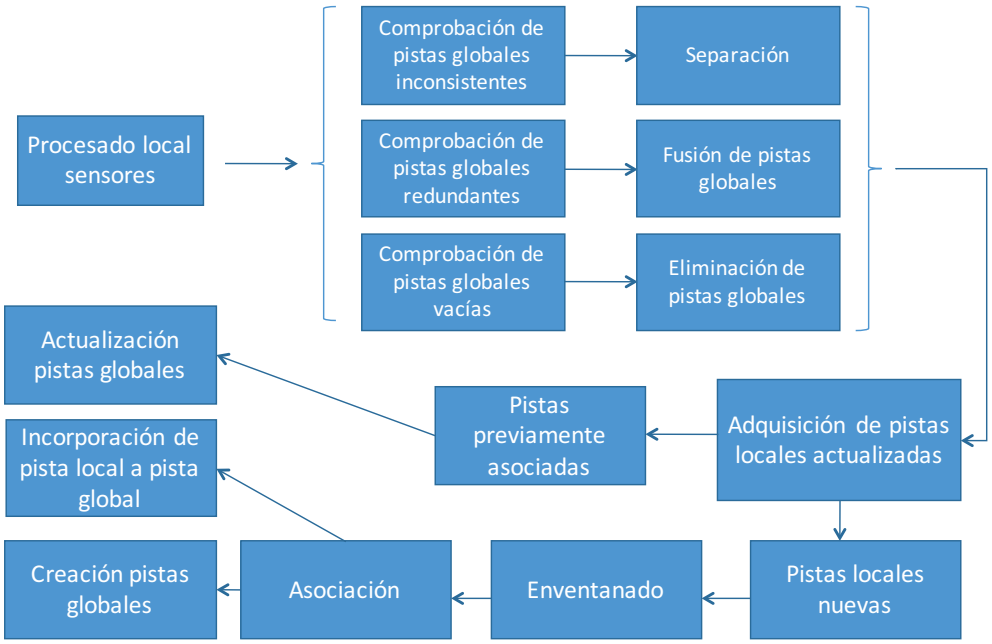


Figura 5.8: Agente Fusión (*FA*) que se encontraría combinando las medidas de posición suministradas por los diferentes sensores como el radar y el AIS. La salida del *FA* consistiría en un conjunto de pistas no redundantes.

Aunque este esquema de fusión sea aparentemente sencillo, incluye múltiples algoritmos estadísticos, de filtrado, de alineación temporal, etc. Podemos encontrar algoritmos de filtrado como el filtro de Kalman (Kalman, 1960), o filtro IMM (Interacting Multiple Model) (Blom and Bar-Shalom, 1988). Algoritmos estadísticos como la distancia de Mahalanobis (De Maesschalck et al., 2000) para calcular la distancia entre pares de posición (cuando modelamos el error en posición con una gaussiana). Y algoritmos de asociación como Munkres (Munkres, 1957), PDA (Probabilistic Data Association) (Bar-Shalom and Tse, 1975), o JPDA (Joint Probabilistic Data Association) (Fortmann et al., 1983). La elección de unos u otros algoritmos de filtrado dependerá del entorno con el que estemos trabajando, del rendimiento esperado, y de las capacidades de cómputo que dispongamos.

Por lo tanto, el Agente Fusión integraría principalmente la lógica descrita en la figura 5.8, que permitiría generar un conjunto de pistas globales no redundantes dentro del sistema. Esta información es suministrada principalmente a los Agentes de Control, que se encargarían de las tareas de coordinación. La información de entrada y salida de este agente quedaría descrita en la tabla 5.3.

| | |
|---|---|
| **Entrada** | |
| **Petición de información** | Generada por cualquier agente de la arquitectura que necesite información de fusión. |
| **Pistas Locales Radar** | Conjunto de pistas locales detectadas por los sensores radar. |
| **Pistas Locales AIS** | Conjunto de pistas locales generadas por las estaciones AIS. |
| **Pistas Locales Visión** | Conjunto de pistas locales generadas por las cámaras PTZ. |
| **Salida** | |
| **Pistas Globales** | Conjunto de pistas globales como resultado de la integración de una o varias pistas locales. Esta información contará con información básica como la posición del barco, velocidad, dimensión, y precisión de la medida. Además cuenta con la información de asociación que indica las relaciones entre las pistas locales de cada sensor y la pista global. |

Tabla 5.3: Información de entrada y salida del Agente Fusión (*FA*) para el entorno marítimo.

### 5.3.3 Agente Control

El Agente de Control (*CA*) es el agente encargado del control automático de las cámaras PTZ del entorno con el fin de satisfacer el objetivo global establecido. Su diseño es muy similar al *CA* definido en la sección 4.3.3 en cuanto a la información de entrada e información de salida, y su relación con el resto de agentes. El objetivo de este agente consistirá en el control de las cámaras de tal modo que en cada momento cada una de las cámaras se encuentre monitorizando un blanco diferente. Si un blanco ya se encuentra cubierto por otra cámara deberá escanear el entorno por las zonas definidas de interés. Para ello deberá utilizar la información fusionada generada por el *FA* con el fin de poder realizar la toma de decisiones sobre las entidades presentes en el entorno.

Para este entorno en concreto, y a modo ilustrativo, se ha desarrollado un sistema de toma de decisiones basado en el Algoritmo de Análisis Jerárquico (AHP) que se describe en la sección 5.3.3.1. En este presentamos un sistema analítico de priorización de monitorización de objetivos basados en la distancia a la cámara, prioridad de las zonas, y la velocidad de la embarcación. La elección de este algoritmo en concreto es que nos permitirá trabajar fácilmente con un experto del dominio que permita ajustar las prioridades de monitorización de los agentes. No obstante, la implementación concreta del *CA* para un entorno real dependerá de los objetivos reales que se pretendan satisfacer, y sería posible contemplar el uso de múltiples técnicas como lógica borrosa, redes bayesianas, algoritmos de clasificación, etc.

La información de entrada y salida con la que contaría este sensor la podemos resumir en la tabla 5.4. Como se puede observar, este agente podría llegar a recibir información de cada

uno de los sensores de manera independiente. En un entorno real posiblemente no se de esta situación, ya que con la información de fusión sería suficiente. Pero de manera experimental, en la sección de evaluación, se valorará el comportamiento de este agente cuando dispone de diferentes fuentes de información. Como pueden ser las cámaras, los sensores radar y AIS, o el sistema de fusión.

| **Entrada** | |
|---|---|
| **Información de coordinación** | Generada por otros agentes homólogos que necesiten coordinarse para solucionar un problema de forma conjunta. |
| **Pistas Locales Radar** | Conjunto de pistas locales detectadas por los sensores radar (a través del *SA*). |
| **Pistas Locales AIS** | Conjunto de pistas locales generadas por las estaciones AIS (a través del *SA*). |
| **Pistas Locales Visión** | Conjunto de pistas locales generadas por las cámaras PTZ (a través del *SA*). |
| **Pistas Globales** | Conjunto de pistas globales generadas por el *FA* tras el proceso de integración de una o múltiples pistas locales de los sensores. |
| **Salida** | |
| **Información de Control** | Control realizado sobre la cámara PTZ que se encuentra gestionando. Este control se generará en función de las detecciones percibidas del entorno y los mecanismos de priorización y coordinación con los que cuente el agente.. |

Tabla 5.4: Información de entrada y salida del Agente Control (*CA*) para el entorno marítimo.

En las siguientes secciones se explica en más detalle los mecanismos de priorización y coordinación que se han contemplado para el diseño de este *CA* particular para el entorno marítimo.

#### 5.3.3.1 Priorización de objetivos

La toma de decisiones para un sistema multi-agente puede ser visto como un proceso cognitivo que permite seleccionar una acción entre las diferentes posibilidades que brinda el entorno. El resultado de un proceso de toma de decisiones es generar una decisión final. En este entorno, las alternativas son las diferentes embarcaciones que es necesario monitorizar, y la elección, la embarcación concreta a ser monitorizada. Siempre hay que sopesar las ventajas y desventajas

en este tipo de problemas, ya que la monitorización de una embarcación determinada prevendrá la monitorización de otras posibles embarcaciones de interés. De este modo, es importante establecer prioridades acerca de las embarcaciones que serán monitorizadas en cada momento.

Hay muchas alternativas que permiten resolver el problema de toma de decisiones con múltiples criterios (Multi-Criteria Decision-Making o MCDM) en los sistemas multi-agente, tal y como se describe en los trabajos presentados en (Boussard et al., 2007), y (An et al., 2008). Para esta aplicación en particular, nosotros hemos seleccionado el proceso de análisis jerárquico (Analytic Hierarchy Process o AHP). El proceso AHP tiene la característica de que es un proceso asequible que permite estructurar el problema de decisión, representar y cuantificar los criterios, relacionar estos criterios con objetivos globales, evaluar las alternativas, etc. De esta forma podríamos llegar a contar con un experto del dominio que ajuste fácilmente los pesos en función de sus prioridades o las condiciones actuales del entorno. De todas formas, la elección del proceso AHP es una forma de ilustrar cómo integrar algunas lógicas que permiten la priorización de objetivos en este tipo de entornos, no como una propuesta de solución general para este tipo de dominios.

En una jerarquía AHP simplificada es necesario seleccionar un objetivo, definir un criterio, y presentar las diferentes alternativas para su evaluación. En este entorno queremos monitorizar una embarcación entre todas las alternativas presentes en el entorno, por lo que este será nuestro objetivo. El criterio de selección y sus prioridades deberían ser acordadas con un experto en el dominio. En este ejemplo en particular hemos escogido a modo ilustrativo criterios basados en la distancia de la embarcación a la cámara, su velocidad, y la prioridad de la zona en la que se encuentra. Esta jerarquía de dos niveles queda representada en la figura 5.9, donde se puede observar el objetivo global de seleccionar una pista del entorno, los diferentes criterios, y el conjunto de alternativas que se presentan en el entorno.

En el proceso AHP, las prioridades de los criterios se establecen definiendo matrices de comparación de pares (Pairwise Comparison Matrix o PCM), donde se establecen los pesos entre los diferentes criterios. Un ejemplo de matriz de comparación de pares queda definido en la matriz (5.1) para este problema concreto, donde se comparan criterios como la distancia de una pista, su velocidad, o la zona de prioridad.

$$
\begin{array}{l}
PairwiseComparisonMatrix(PCM) = \\
\end{array}
\begin{array}{l}
\\
DISTANCE \\
SPEED \\
ZONE
\end{array}
\begin{array}{c}
\begin{array}{ccc}
DISTANCE & SPEED & ZONE
\end{array} \\
\begin{pmatrix}
1 & 2 & \frac{1}{2} \\
\frac{1}{2} & 1 & \frac{1}{3} \\
2 & 3 & 1
\end{pmatrix}
\end{array}
\tag{5.1}
$$

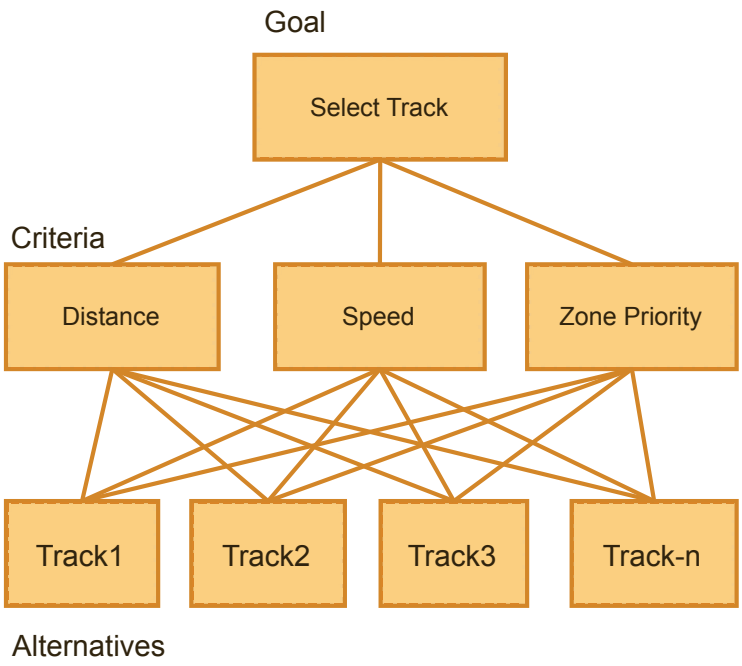


Figura 5.9: Proceso de Análisis Jerárquico (AHP) simplificado que permitirá a los agentes tomar decisiones acerca de la embarcación a ser monitorizada en cada momento. Para ello se basará en criterios como la distancia de la embarcación a la cámara, su velocidad, y la prioridad de la zona en la que se encuentra.

Los valores de una matriz de comparación de pares se establecen de acuerdo a las preferencias que tenga el operador sobre cada uno de los criterios. Para ello que establecen valores de prioridad entre pares con valores entre 1 y 9. El 1 indica la misma prioridad, y el 9 indica una prioridad absolutamente superior (Saaty, 1990). El proceso AHP además permite evaluar la inconsistencia de los criterios calculando un índice de consistencia (CI) basado en el valor de Eigen para luego generar un ratio de consistencia (CR). Un CR es aceptable cuando está por debajo del 10 %. Estos indicadores son especialmente útiles cuando hay múltiples criterios involucrados y es difícil ajustar los pesos cualitativamente. El CR para el ejemplo propuesto se encuentra por debajo del 1 %, por lo que podríamos considerar que los criterios son consistentes.

El proceso AHP generará un vector de prioridad de criterios en base a la matriz de comparación de pares establecidos de forma manual por el operador. Esto es básicamente un cálculo que genera el peso absoluto de cada criterio en función de los pesos relativos propuestos por el operador. Para ello es necesario calcular el vector Eigen normalizado de la matriz (Saaty, 2003), que se podrá utilizar como vector de prioridades de criterios. A modo de ejemplo se presenta en (5.2) el resultado del cálculo realizado sobre la matriz de pesos (5.1). Se puede observar cómo el criterio zonal obtiene una mayor prioridad que la distancia o velocidad de la

embarcación.

$$CriteriaPriorityVector(CPV) =$$

$$\begin{matrix} DISTANCE \\ SPEED \\ ZONE \end{matrix} \begin{pmatrix} 0,297 \\ 0,163 \\ 0,540 \end{pmatrix} \tag{5.2}$$

En el tercer nivel de AHP se presentan las alternativas. Estas también deberían ser comparadas en una matriz de comparación de pares por cada criterio para luego calcular sus vectores de prioridad. Con el fin de simplificar el problema de asignar prioridades cualitativas (del 1 al 9) en base a información cuantitativa como la distancia, velocidad, zona, etc, calcularemos directamente los vectores de prioridades. En este caso particular se asume que la distancia tiene más peso cuanto más pequeña es, ya que la embarcación está más cerca de la cámara. Para ello el peso de cada alternativa se calcula según la fórmula descrita en (5.4). Por el contrario, la velocidad y la prioridad zonal tendrán más peso cuanto mayor sean, y el vector de prioridades se calculará según se describe en (5.3).

$$weight_i = \frac{value_i}{\sum_{j=1}^{n} value_j} \tag{5.3}$$

$$weight_i^{'} = \frac{(\sum_{j=1}^{n} value_j) - value_i}{\sum_{j=1}^{n} value_j} \tag{5.4}$$

Estos vectores de prioridad calculados para cada criterio se integran en una matriz que representaría las prioridades de cada alternativa, tal y como se muestra se muestra en (5.5). En esta matriz cada columna representa un criterio, mientras que cada fila representa cada una de las alternativas, que en nuestro caso serían las embarcaciones que estamos evaluando. Por lo tanto, un elemento cualquiera de la matriz $APM_{i,j}$ especifica el peso que tiene la embarcación

de la fila de la matriz $i$ para el criterio de la columna $j$.

$$AlternativesPriorityMatrix(APM) = \begin{array}{c} \\ Alternative_1 \\ Alternative_2 \\ \\ Alternative_n \end{array} \begin{array}{c} \begin{array}{ccc} DISTANCE & SPEED & ZONE \end{array} \\ \begin{pmatrix} DistanceWeight_1 & SpeedWeight_1 & ZoneWeight_1 \\ DistanceWeight_2 & SpeedWeight_2 & ZoneWeight_2 \\ \dots & \dots & \dots \\ DistanceWeight_n & SpeedWeight_n & ZoneWeight_n \end{pmatrix} \end{array} \tag{5.5}$$

Con la matriz de prioridad de las alternativas $APM$ y el vector de prioridad de criterios $CPV$, podemos conseguir un vector de prioridades que represente la prioridad global de cada una de las alternativas. Para ello basta con multiplicar la matriz $APM$ con el vector $CPV$, tal y como se define en (5.6).

$$TracksPriorityVector(TPV) = APM * CPV \tag{5.6}$$

El vector resultante representará la prioridad de cada pista para ser monitorizada. Una especie de clasificación general entre todas las pistas evaluadas. De esta forma, elegir la pista con más peso, será equivalente a seleccionar la pista con más prioridad de acuerdo a las preferencias establecidas por el operador en el primer paso. Cada Agente de Control, por tanto realizará este proceso periódicamente en función de la información de pistas recibidas por el Agente Fusión. De este modo cada agente podrá seleccionar aquellas pistas de más interés para el operador.

La selección de estas pistas por parte de cada $CA$ digamos que se realiza de una forma un tanto ambiciosa. Esto podría generar conflictos entre los agentes cuando una de las pistas se convierte en la más importante para dos o más $CA$. Para ello, los agentes implementarán un algoritmo de coordinación que se describe en la siguiente sección.

#### 5.3.3.2 Resolución de conflictos

Como cada uno de los $CA$ que se encuentran percibiendo el entorno calcula las prioridad de monitorización de cada pista de forma independiente, puede existir la posibilidad de que más de un agente se interese en monitorizar la misma pista. Esto reduciría la eficiencia del sistema multi-agente ya que nos encontraríamos con varias cámaras realizando una monitorización redundante. Podría ser más interesante que una de las cámaras se encargase de la monitorización, mientras que el resto visualizan otras partes de interés del entorno.

Para poder llevar a cabo este reparto de tareas serán los propios agentes los que apliquen técnicas de colaboración para solucionar sus conflictos de manera distribuida. Existen múltiples trabajos de investigaciones realizadas en el ámbito de la toma de decisiones colaborativas con múltiples criterios, pero por simplificar el diseño de esta aplicación, se ha optado por un sistema de pujas entre agentes (Maes et al., 1999).

El algoritmo de coordinación comienza cuando un *CA* selecciona una pista a monitorizar. Esta selección se notifica al resto de *CA* para que tengan constancia de las intenciones del agente. Esta notificación es tentativa, lo que quiere decir que el *CA* que notifica su intención debe esperar un período de tiempo razonable para que el resto de *CA* puedan presentar sus quejas. Si ningún otro *CA* se opone a la intención de monitorización pasado el período de espera, el *CA* comenzará inmediatamente con la monitorización de la pista que había seleccionado.

Por el contrario, si algún otro *CA* del entorno encuentra alguna discrepancia, ya sea porque ya se encuentra monitorizando esa pista, o ha seleccionado la misma, es necesario iniciar un proceso de coordinación. Este proceso de coordinación consistirá básicamente en que cada agente interesado en monitorizar la pista comunica al resto de agentes la prioridad de su pista, a modo de puja. El agente ganador será aquél cuya prioridad de monitorización sea mayor.

Cada uno de los agentes mantendrá un histórico de pujas perdidas con el fin de evitar un proceso de negociación continuo. Una pista que tenga la mayor prioridad en el instante $t$, es muy probable que la siga manteniendo en el instante $t+1$. Esto podría disparar continuamente los procesos de notificación tentativa y negociación. Para solucionar esto, el agente que ha perdido la puja guardará un registro del resultado, de tal modo que en la siguiente evaluación desechará aquellas pistas que ya haya intentado monitorizar sin éxito. Es obvio que el estado del entorno en este tipo de entornos es muy cambiante, lo que podría suponer un cambio en los resultados de las pujas ya realizadas. Por este motivo los agentes limpiarán cada registro pasado un determinado intervalo de tiempo, permitiendo así volver a iniciar el proceso de puja en caso de que fuera necesario.

## 5.4 Simulador de entorno de vigilancia marítimo

Evaluar el rendimiento o adecuación de un sistema multi-agente controlando un conjunto de cámaras PTZ en un entorno marítimo real puede llegar a ser una tarea muy complicada. Hay múltiples sensores que requieren de una gran infraestructura normalmente controlada por entidades gubernamentales o empresas. Sería muy difícil conseguir permisos especiales para experimentar sobre este tipo de entornos, la información capturada suele ser de carácter confidencial, y no es habitual disponer de un entorno de vigilancia marítima cercano. Además podemos encontrarnos con otro problema añadido, que consiste en la repetibilidad de los experimentos. Cuando se diseña un sistema que puede incorporar diferentes técnicas, algoritmos y ajustes, es importante poder compararlos bajo las mismas condiciones del entorno, sobre todo en las primeras etapas del diseño y configuración. Esto es prácticamente inviable en un entorno tan cambiante como lo es el entorno de vigilancia marítima, y más si consideramos la gestión autónoma de cámaras PTZ.

Por este motivo se pensó en evaluar este tipo de entorno a través de un entorno marítimo simulado. Con el desarrollo de diferentes proyectos de investigación con empresas A.1, ya se diseñó una herramienta que permitía diseñar y simular trayectorias de embarcaciones. Esto permitía generar escenarios con un conjunto de particularidades para evaluar los diferentes algoritmos de fusión que estábamos desarrollando. Esto sirvió como base para el desarrollo de esta tesis, donde se extendió la herramienta para no sólo contemplar las trayectorias de las embarcaciones, si no también simular radares, AIS y cámaras PTZ. Además se integraron los agentes de más bajo nivel del diseño descrito en esta aplicación, como el Agente Sensor, el Agente Fusión, y el Agente Control.

De esta forma podremos evaluar el comportamiento del sistema multi-agente en un entorno simulado que contenga todos los elementos necesarios. Para realizar el simulador se diseño una aplicación multi-plataforma basada en JavaFx, cuyo componente principal es una vista de mapa. Sobre este mapa, proporcionado por Google Maps, se visualizan todos los elementos simulados como las embarcaciones, los sensores, y determinados elementos geográficos como polígonos, polilíneas, etc. La ventana principal de la aplicación se puede observar en la figura 5.10.

La descripción del diseño de esta herramienta, así como la especificación de los algoritmos de cálculo de trayectorias, simulación de radares, simulación del campo de visión de la cámara, etc., queda fuera del alcance de la tesis. Pero debido a la utilidad de esta aplicación para diferentes proyectos y entornos, se está trabajando en su publicación en algún repositorio de código fuente abierto. Por el momento contamos con un canal de Youtube [1] donde se pueden ver algunas de las características más importantes de la aplicación, que se describen a continuación.

[1] https://www.youtube.com/user/GIAAUC3M

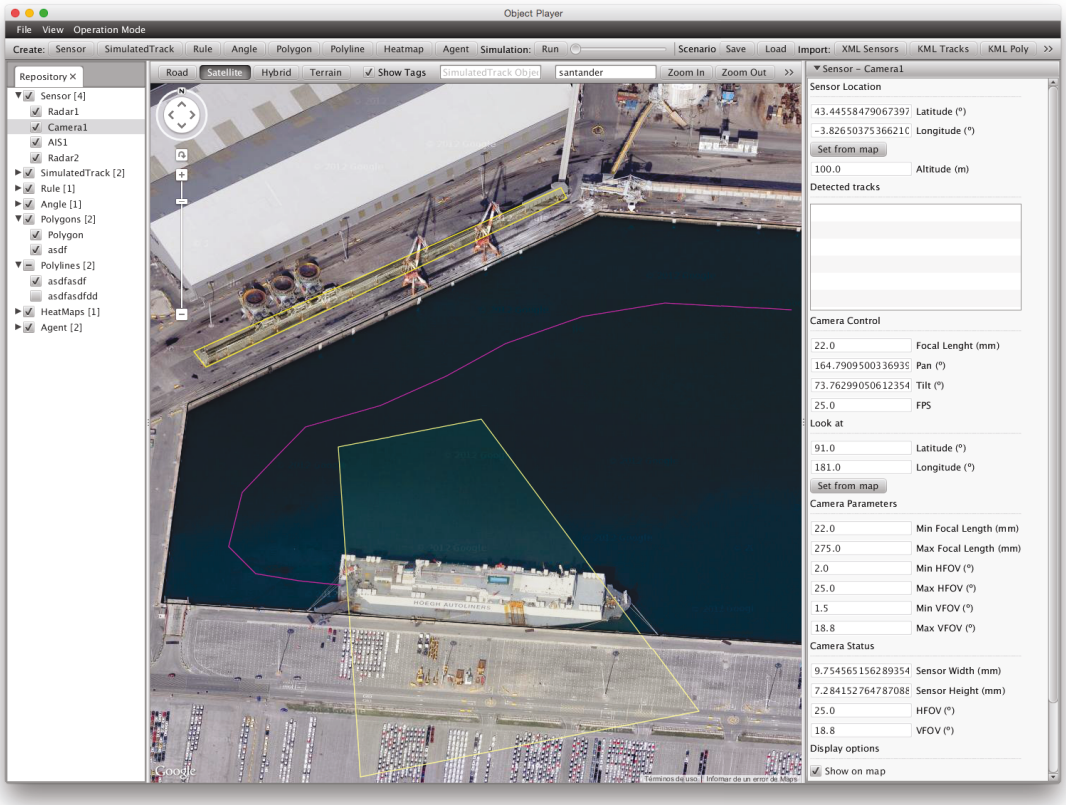


Figura 5.10: Pantalla principal del simulador de entornos marítimos. Desde esta ventana se pueden configurar la mayoría de los elementos de simulación como las embarcaciones, sensores, agentes, etc. Es posible monitorizar todos los elementos y su simulación en tiempo real sobre los mapas de Google Maps.

### 5.4.1 Embarcaciones

El simulador permite incorporar embarcaciones simuladas en una posición determinada. Adicionalmente permitirá definir una trayectoria que permitirá simular el movimiento del barco a diferentes velocidades. La trayectoria se describe como un conjunto de puntos paso, y para cada punto de paso se define su velocidad, velocidad angular, e incluso el tiempo de parada si queremos simular la parada y el posterior arranque de un barco. Esto nos permitirá definir maniobras complejas con diferentes velocidades, aceleraciones y giros.

Una vez iniciada la simulación cada embarcación se moverá de acuerdo a la trayectoria que esta tenga definida. Esto nos permitirá diseñar escenarios complejos con múltiples embarcaciones interactuando y desplazándose por el entorno. Este tipo de herramienta ha sido utilizado en determinados proyectos de investigación con empresas, principalmente para crear escenarios específicos para la fusión de datos donde era necesario cubrir diversos aspectos como el cruce de trayectorias a diferentes distancias, trayectorias que van paralelas, trayectorias que entran o


Figura 5.11: Simulador de trayectorias de embarcaciones. Permite definir un conjunto de puntos de paso en los que se puede especificar la velocidad y velocidad angular. El simulador calculará automáticamente las aceleraciones necesarias entre los puntos de paso para desplazar la embarcación.

salen de la cobertura de un radar, embarcaciones que llegan a puerto, etc.

Los parámetros que se pueden definir en las embarcaciones se describen a continuación.

**Posición (WGS84)**

Define la posición inicial de la embarcación. Se define con la posición en coordenadas WGS84 (latitud y longitud).

**Dimensiones (m)**

Dimensiones físicas de la embarcación en metros.

**Transpondedor AIS**

Es posible equipar a la embarcación con un transpondedor AIS que interactuará con las estaciones AIS simuladas que se describirán más adelante. Permite definir el identificador de la embarcación (MMSI), y el ajuste de posición diferencial o no, que reducirá la incertidumbre en la posición.

**Trayectoria**

La trayectoria se define como un conjunto de puntos de paso en coordenadas WGS84, y

por cada punto de paso se puede especificar la altura (por si queremos definir trayectorias en entornos aéreos), la velocidad de paso (que permitirá calcular las aceleraciones de punto a punto), la velocidad angular (para calcular el radio de giro en función de la velocidad), y por último un retardo (por si la velocidad del punto de paso es 0, poder simular una parada).

Se puede observar un ejemplo de la definición de una trayectoria con este simulador en la figura 5.11. Podemos observar una embarcación que cuenta con una trayectoria definida por cuatro puntos de paso. Podemos comprobar que en los tres puntos de paso intermedios aparece un círculo que describe el radio de giro que realizará la embarcación. Esto se calcula en función de la velocidad y velocidad angular definida en cada punto de paso, y el simulador lo tendrá en cuenta para realizar trayectorias con giros más o menos suaves. En determinados proyectos, esta característica ha permitido probar el funcionamiento de los algoritmos de filtrado en las maniobras, y así evaluar sus parámetros, modos de funcionamiento, etc.

### 5.4.2 Sensores Radar

Se pueden configurar sensores virtuales de tipo radar que permitirán simular las detecciones de las diferentes embarcaciones configuradas en el entorno. Este tipo de sensores permite escanear el entorno en busca de objetivos de forma periódica en función del período de rotación de la antena. En un entorno de vigilancia marítimo es común encontrar más de un sensor de este tipo cubriendo diferentes áreas de cobertura, principalmente debido a que estos sensores suelen contar con múltiples oclusiones en función de la orografía concreta del entorno. De este modo, en el simulador podremos configurar múltiples estaciones radar cubriendo diferentes áreas.

El simulador del sensor radar no pretende ser totalmente realista, ya que un sistema radar real es un elemento complejo de ajustar, procesar, y además cuenta con multitud de parámetros que intervienen para su correcto funcionamiento. En este entorno trataremos de definir características básicas sobre un sistema de radar, como su posición, períodos de rotación y precisiones, de tal forma que las detecciones finales del entorno simulado se parezcan a lo que un sistema radar real podría llegar a proporcionar. Pero sin llegar a simular las físicas y parámetros de bajo nivel por los que se rigen este tipo de sensores. Los parámetros configurables de un radar simulado se describen a continuación.

**Posición (WGS84)**

Define la posición de la estación radar y su altura. Se define con la posición en coordenadas WGS84 (latitud y longitud).

**Período de rotación (ms)**

Define el período de rotación de la antena en milisegundos. Esto afectará principalmente al período de refresco de las embarcaciones detectadas.

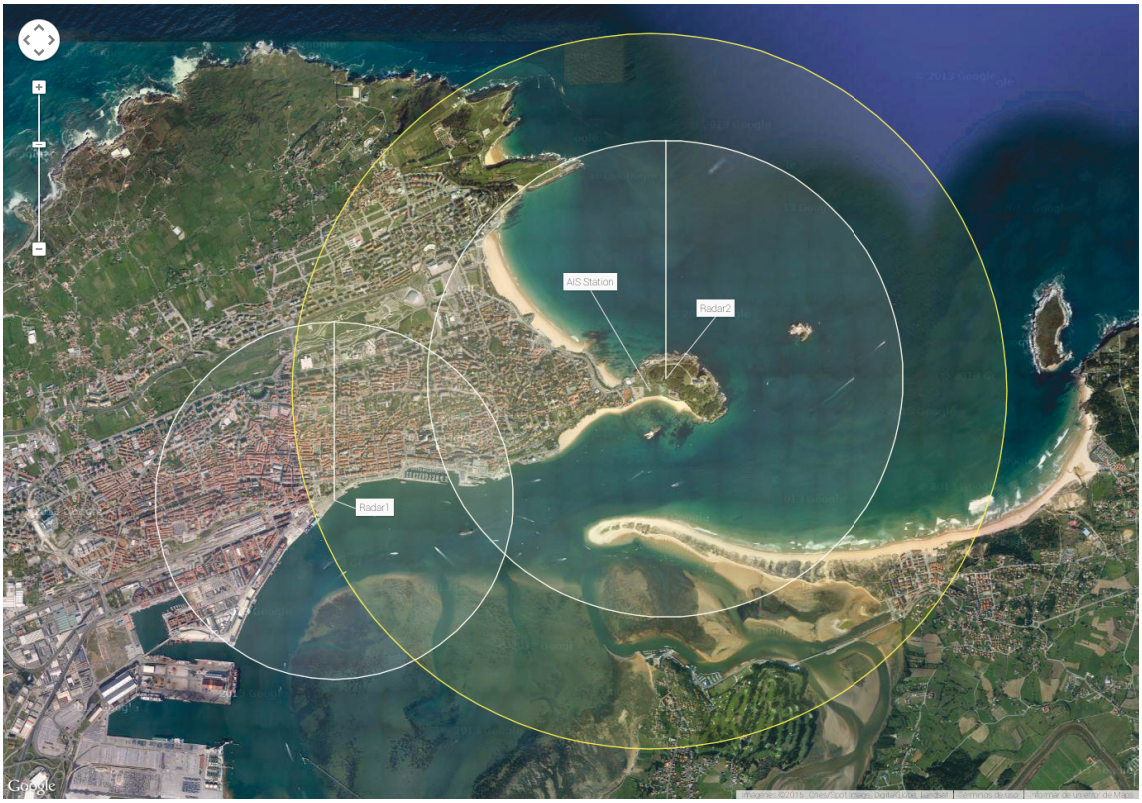


Figura 5.12: Representación de un conjunto de sensores simulados. En la figura podemos observar dos radares y una estación AIS. Cada uno de estos sensores cuenta con una zona de cobertura que limitará las detecciones de blancos en el entorno.

**Alcance máximo (m)**

Permite definir el alcance máximo de la estación en metros. Normalmente este parámetro viene determinado en los entornos reales por el tipo de antena, la configuración de los pulsos de energía, etc.

**Distancia mínima (m)**

Define la distancia mínima de detección.

**Precisión en Distancia y Azimut (m)**

La detección generada por cada radar normalmente tiene asociada un margen de error que depende directamente de la resolución en distancia y azimut del radar. Este error variará principalmente en función de la distancia, ya que las celdas de resolución se hacen más grandes cuanto más lejano se encuentra el blanco. También influirán parámetros como el ancho de pulso, período de rotación, etc. En este caso también simplificamos el cálculo de la precisión y suponemos un error constante en distancia y azimut durante la operación del radar simulado.

Una vez configurados estos parámetros es posible simular el funcionamiento del radar junto con las embarcaciones definidas. En la simulación, el radar comenzará a girar según el tiempo configurado en el período de rotación. Las detecciones de las embarcaciones se generan cuando el pulso del radar simulado pasa por la ubicación de la embarcación, siempre y cuando se encuentre dentro de su cobertura. Estas detecciones generan información muy similar a la que generaría un radar real, como las coordenadas polares de la embarcación respecto a la antena, su posición en coordenadas WGS84, el error en distancia y azimut, identificador de la pista, etc. Toda esta información la podemos emplear como entrada en un sistema de fusión de información como el descrito en el Agente Fusión.

Podemos encontrar una representación de un conjunto de sensores radar, junto con una estación AIS en la figura 5.12. Se puede observar la ubicación de cada una de las estaciones así como sus rangos de cobertura. En estos rangos de cobertura estos sensores serán capaces de generar detecciones en función de los períodos de rotación de la antena.

### 5.4.3 Estaciones AIS

Las estaciones AIS son sensores totalmente diferentes a los radares, ya que estos no cuentan con sofisticados sistemas de detección basados en ondas electromagnéticas. En este caso una estación AIS consta básicamente de una antena de radio VHF (Very High Frequency) con la que escucha los mensajes emitidos por las embarcaciones cercanas. Los mensajes son generados automáticamente por las embarcaciones, que normalmente cuentan con un sistema de posicionamiento por satélite, así como un transpondedor para poder emitir los mensajes.

Estos mensajes se encuentran codificados de acuerdo a un protocolo estándar de comunicación que define alrededor de 27 tipos de mensaje diferentes. Estos mensajes pueden contener diversa información, como un identificador único de la embarcación (MMSI), la posición y velocidad del barco, sus dimensiones, el nombre, el puerto de destino de la embarcación, el tiempo estimado de llegada, el estado en el que se encuentra, etc. Los mensajes relacionados con la posición y velocidad del barco (llamados mensajes dinámicos) se envían periódicamente en función de la dinámica actual del barco. Un barco que vaya más rápido está obligado a aumentar la frecuencia de actualización de sus mensajes dinámicos. Mientras que un barco que se encuentra anclado en puerto podría mandar actualizaciones cada 180 segundos. Los mensajes estáticos, que contienen información como el nombre del barco y su destino, se envían con una frecuencia mucho menor.

Por lo tanto una estación AIS únicamente se encargará de recibir estos mensajes, decodificarlos y proporcionarlos al sistema de vigilancia marítima. Con esta información el operador puede observar con bastante nivel de detalle la situación actual del entorno, independientemente del uso de sensores radar. En este simulador permitimos también simular un conjunto de estaciones AIS que básicamente definirán su posición y rango de cobertura. Las embarcaciones simuladas

que equipen un transpondedor AIS y se encuentren dentro de la cobertura de una estación AIS serán detectadas. Las embarcaciones además simulan la frecuencia de actualización variable en función de la velocidad actual, aunque sólo serán simulados los mensajes dinámicos que informan sobre la posición y velocidad. Los parámetros configurables de una estación AIS se describen a continuación.

**Posición (WGS84)**

Define la posición de la estación AIS y su altura. Se define con la posición en coordenadas WGS84 (latitud y longitud).

**Rango de cobertura (m)**

Define el rango de cobertura máximo de la estación AIS en metros. En un entorno real esto depende de múltiples factores como de la calidad y potencia del transpondedor ubicado en la embarcación, así como de la sensibilidad y visibilidad de la antena.

Una vez iniciada la simulación, del mismo modo que los radares comenzarán a generar detecciones, las estaciones AIS comenzarán a capturar los mensajes emitidos por las diferentes embarcaciones. Estos mensajes cuentan con información de posición en coordenadas WGS84, así como velocidad y rumbo. Esta información se podrá utilizar como entrada para el proceso de fusión, que permitirá integrar las detecciones realizadas por los sensores radar y las estaciones AIS. En la figura 5.12 se puede observar un ejemplo de una estación AIS desplegada en el entorno junto con dos sensores radar. El alcance de una estación AIS suele ser de varias ordenes de magnitud superior a la cobertura de un radar, por lo que una única estación puede cubrir fácilmente cientos de kilómetros a la redonda.

### 5.4.4 Cámaras PTZ

Las cámaras PTZ son otro de los elementos más comunes en un sistema de vigilancia marítima. Estas permiten al operador centrar la atención en determinados aspectos o elementos de interés del entorno. Son una fuente de información necesaria para la monitorización de procesos como operaciones de remolque, maniobras de atraque, detección de amenazas, etc. Este tipo de cámaras, normalmente diseñadas específicamente para este tipo de entornos, se encontrarían ubicadas en lugares estratégicos desde los que poder monitorizar las áreas de mayor importancia del entorno.

Del mismo modo que el resto de sensores, desde el simulador se podrá configurar un conjunto de cámaras PTZ que se encuentre monitorizando el entorno. Estas cámaras pueden actuar de forma similar a como lo haría una cámara real, ya que se simulan los parámetros ópticos de la cámara para calcular el campo de visión de la cámara en cada momento. Además simula los controles de movimiento de la cámara, pudiendo realizar cambios en su orientación


Figura 5.13: Ejemplo de dos cámaras PTZ simuladas en una zona de un puerto marítimo. El simulador puede generar el campo de visión teórico de la cámara en función de los parámetros ópticos configurados en la cámara. Además permite simular la detección de los blancos que se encuentren dentro de su campo de visión.

en tiempo real. También se ha simulado el procesado de las imágenes, que permite detectar los blancos del entorno que entran dentro de su campo de visión, de forma similar a como lo haría cualquier otro sensor, o la implementación de ejemplo que se vio en el capítulo 4. De esta forma las cámaras pueden considerarse como una fuente de información más dentro del entorno simulado, pero estas además permiten ser configuradas en tiempo real para ajustar el campo de visión en función de las necesidades.

Para simular este tipo de cámaras es necesario definir las propiedades del sensor óptico que vamos a simular. De esta forma podemos tomar la posición de la cámara y su orientación, y realizar una proyección simulada del campo de visión sobre el entorno. La mayoría de los parámetros que es necesario ajustar es posible encontrarlos en las especificaciones de los fabricantes. De hecho, para nuestras simulaciones hemos ajustado los parámetros de las cámaras para un modelo concreto del fabricante FLIR, especializado en la fabricación de todo tipo de sensores de visión, entre otros cámaras PTZ para entornos marítimos. Los parámetros que son necesarios definir se describen a continuación:

**Posición (º)**

Define la posición de la cámara y su altura. Se define con la posición en coordenadas WGS84 (latitud y longitud en grados).

**Distancia focal mínima y máxima (mm)**

Especifica la distancia focal mínima y máxima en milímetros que tiene la cámara. Esto permite definir la capacidad de zoom de la cámara.

**Campo de visión horizontal (º)**

Define el campo de visión horizontal de la cámara en grados. Para ello se proporcionará el valor mínimo y máximo del HFOV (Horizontal Field Of View).

**Campo de visión vertical (º)**

Define el campo de visión vertical de la cámara en grados. Es necesario proporcionar el valor mínimo y máximo del VFOV (Vertical Field Of View).

**Dimensiones del sensor (mm)**

Dimensiones del sensor (ancho y alto) en milímetros. Normalmente estos parámetros podrán calcularse en función del VFOV y HFOV.

**Fotogramas por segundo (fps)**

Aunque la cámara simulada no genere imágenes sintéticas del entorno, es posible simular la frecuencia de actualización de la cámara, y por tanto la frecuencia de actualización de las detecciones que este tipo de sensor pueda generar.

Los elementos que se pueden ajustar en tiempo real para simular el control de la cámara son similares a los que permiten ajustar cualquier cámara PTZ real:

**Distancia focal (mm)**

Ajustando la distancia focal podemos cambiar el nivel de zoom de la cámara.

**Posición horizontal (º)**

Permite orientar la cámara en el eje horizontal.

**Posición vertical (º)**

Permite orientar la cámara en el eje vertical.

Se puede observar una representación de este tipo de sensores en la figura 5.13, donde podemos observar dos cámaras en una zona de puerto. Estas cámaras podrán ser utilizadas en tiempo real para simular sistemas de vigilancia autónomos que gestionen y coordinen las cámaras para satisfacer diferentes tareas de vigilancia.

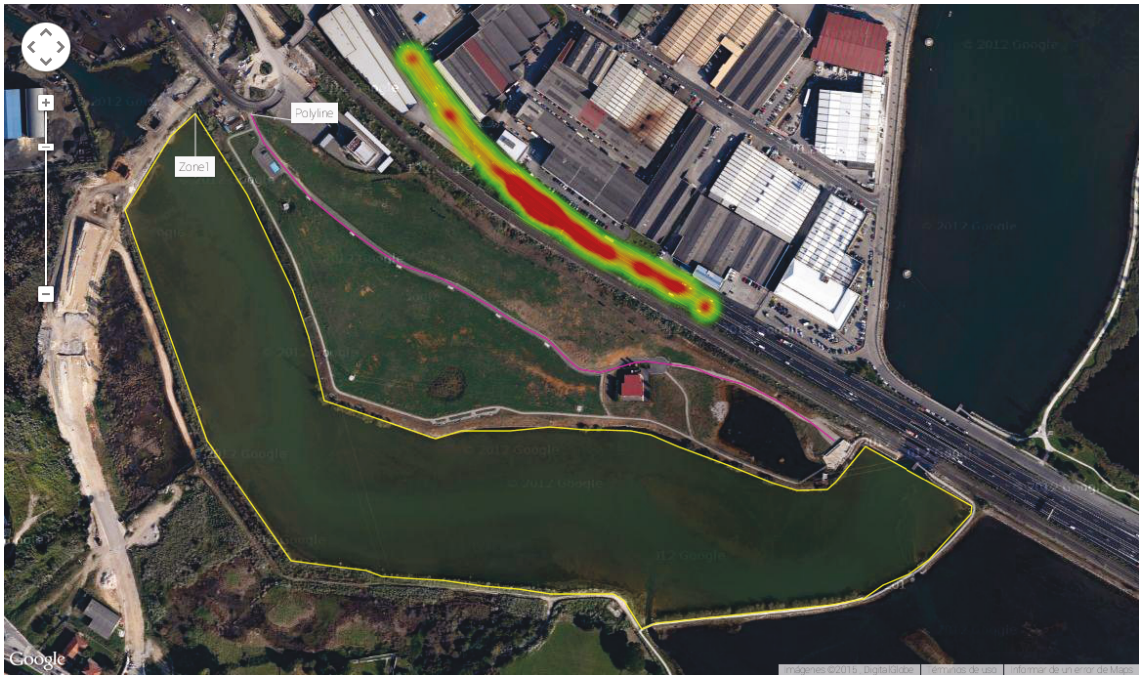


Figura 5.14: Ejemplo de herramientas geográficas incorporadas en el simulador. Es posible definir polígonos genéricos, polilíneas, mapas de calor, calcular distancias, ángulos, etc. Especialmente útil para el diseño y evaluación de escenarios de vigilancia marítima.

### 5.4.5 Elementos geográficos

El simulador dispone de múltiples herramientas adicionales para poder definir elementos geográficos, realizar mediciones entre elementos, calcular distancias, crear mapas de calor, etc. Esto permite ayudar en el diseño de escenarios complejos, ya que es posible definir zonas geográficas para cambiar la configuración de un determinado algoritmo, definir zonas con prioridades, describir rutas de paso, etc.

La mayoría de estas herramientas se han creado como soporte en la depuración de sistemas de fusión de información, ya que aparte de la simulación, es una herramienta de visualización de escenarios. En diferentes proyectos con empresas donde hemos desarrollado sistemas de fusión, ha sido necesario poder visualizar claramente la salida de nuestro sistema. Principalmente para poder evaluar el rendimiento y consistencia de los algoritmos, el resultado de la fusión, etc. Esta herramienta define un protocolo de paso de mensajes, ya sea a través de fichero o una conexión TCP/IP, para poder monitorizar elementos geográficos en tiempo real. Se puede ver en la figura 5.14 un conjunto de elementos geográficos como polígonos, polilíneas y mapas de calor que pueden ser de utilidad para el diseño y evaluación.

### 5.4.6 Agentes

En las secciones anteriores hemos podido descubrir algunos de los elementos que se pueden simular dentro del entorno de vigilancia marítima. Desde embarcaciones que pueden describir diferentes trayectorias y dinámicas, hasta desplegar sensores por el entorno como radares, estaciones AIS y cámaras PTZ. Esto sensores son capaces de generar información simulada en tiempo real sobre el estado del entorno, por lo que es necesario poder simular la lógica de gestión que estamos describiendo a lo largo de la tesis.

En este sentido, el simulador también permite desplegar un conjunto de agentes simulados, que integran las lógica descrita en el diseño del sistema multi-agente. En concreto permite simular los tres agentes más importantes de la arquitectura: el Agente Sensor, el Agente Fusión, y el Agente Control. Con estos tres tipos de agente podemos simular la gestión multi-sensor y evaluar el comportamiento del sistema multi-agente para el objetivo que nos hemos propuesto en esta aplicación. Los agentes que se podrán desplegar en el simulador se describen a continuación:

**Agente Sensor Radar**

Este agente cumplirá el rol que se describe en la sección 5.3.1. En concreto se encargará de percibir la información generada por un sensor radar que se le asigne. Esta información será transmitida a otros agentes de la arquitectura que puedan necesitar esta información.

**Agente Sensor AIS**

Este agente cumplirá el rol que se describe en la sección 5.3.1. En concreto se encargará de percibir la información generada por una estación AIS que se le asigne. Esta información será transmitida a otros agentes de la arquitectura que puedan necesitar esta información.

**Agente Sensor PTZ**

Este agente cumplirá el rol que se describe en la sección 5.3.1. En concreto se encargará de percibir la información generada por una cámara PTZ que se le asigne. Esta información será transmitida a otros agentes de la arquitectura que puedan necesitar esta información.

**Agente Fusión**

Este agente se comportará según se define en la sección 5.3.2. Recibirá la información generada por los agentes de sensor Radar, AIS, y cámara PTZ para generar una información fusionada. Esta información estará disponible para el resto de agentes de la arquitectura que necesiten acceder a la información de fusión, como el Agente de Control.

**Agente Control PTZ**

El agente de control implementa la lógica de gestión descrita en la sección 5.3.3. Este agente de control de cámaras PTZ se coordinará con sus homólogos para realizar una gestión coordinada de las cámaras PTZ. Para ello podrá recibir la información del entorno desde diferentes fuentes de información, ya sea únicamente a través de las cámaras, de

los sensores, o del sistema de fusión. En la experimentación se evaluará el rendimiento del sistema utilizando estas tres fuentes de información.

## 5.5 Experimentación

En esta sección se evaluará el diseño del sistema multi-agente que integra la fusión de información como fuente de datos para la coordinación de un conjunto de cámaras PTZ en el entorno de vigilancia marítimo. Esta experimentación nos permitirá evaluar la adecuación del sistema multi-agente propuesto en la tesis para su aplicación en este tipo de entornos en particular. Para ello, se usará el simulador descrito en la sección 5.4, que integra el diseño del sistema multi-agente definido en el apartado 5.3. Se evaluará también la adecuación y la necesidad de integración de un sistema de fusión de información como elemento indispensable para mejorar el rendimiento del sistema.

Primero se detallará el escenario sobre el que se desarrollarán los experimentos, seguido de dos evaluaciones que se realizarán en términos de redundancia de cámaras monitorizando los diferentes elementos del entorno, y otro en términos del tiempo que han estado empleando las cámaras realizando las diferentes tareas de seguimiento.

### 5.5.1 Descripción del escenario simulado

El escenario propuesto se desarrolla sobre el puerto marítimo de Santander. Está compuesto por dos sensores radar y una estación AIS. De forma adicional contamos con tres cámaras PTZ ubicadas en posiciones similares a las de los sensores. Cada cámara PTZ será gestionada por un único agente de control tal y como se describe en el diseño del sistema multi-agente. El comportamiento por defecto del agente será mover la cámara alrededor de una zona de monitorización definida hasta que en el escenario aparezca alguna embarcación que pueda ser monitorizada. El objetivo global del sistema multi-agente es la monitorización de las diferentes embarcaciones de manera no redundante. Es decir, una embarcación debería ser monitorizada únicamente por una sola cámara, de esta forma se maximiza el número de embarcaciones monitorizadas en el entorno.

Podemos encontrar una representación del escenario concreto en la figura 5.15. En esta se pueden observar los dos radares (Radar1, y Radar2) desplegados en la zona portuaria. Su cobertura queda representada por la circunferencia de color blanco. Por otro lado contamos con una única estación AIS (AISStation), con una cobertura bastante mayor que se describe con una circunferencia de color negro. Sobre este escenario hay además desplegadas tres cámaras PTZ (Camera1, Camera2, Camera3) que tienen la misma ubicación que los tres sensores radar y AIS del entorno. Estas tienen una limitación de visibilidad máxima de unos 3000 metros. Se puede observar una representación de su campo de visión delimitado por un polígono de color azul. También se puede observar determinadas áreas predefinidas de monitorización en color rojo. Estas áreas representan zonas que las cámaras tienen que observar en caso de no encontrar barcos en su zona de cobertura.

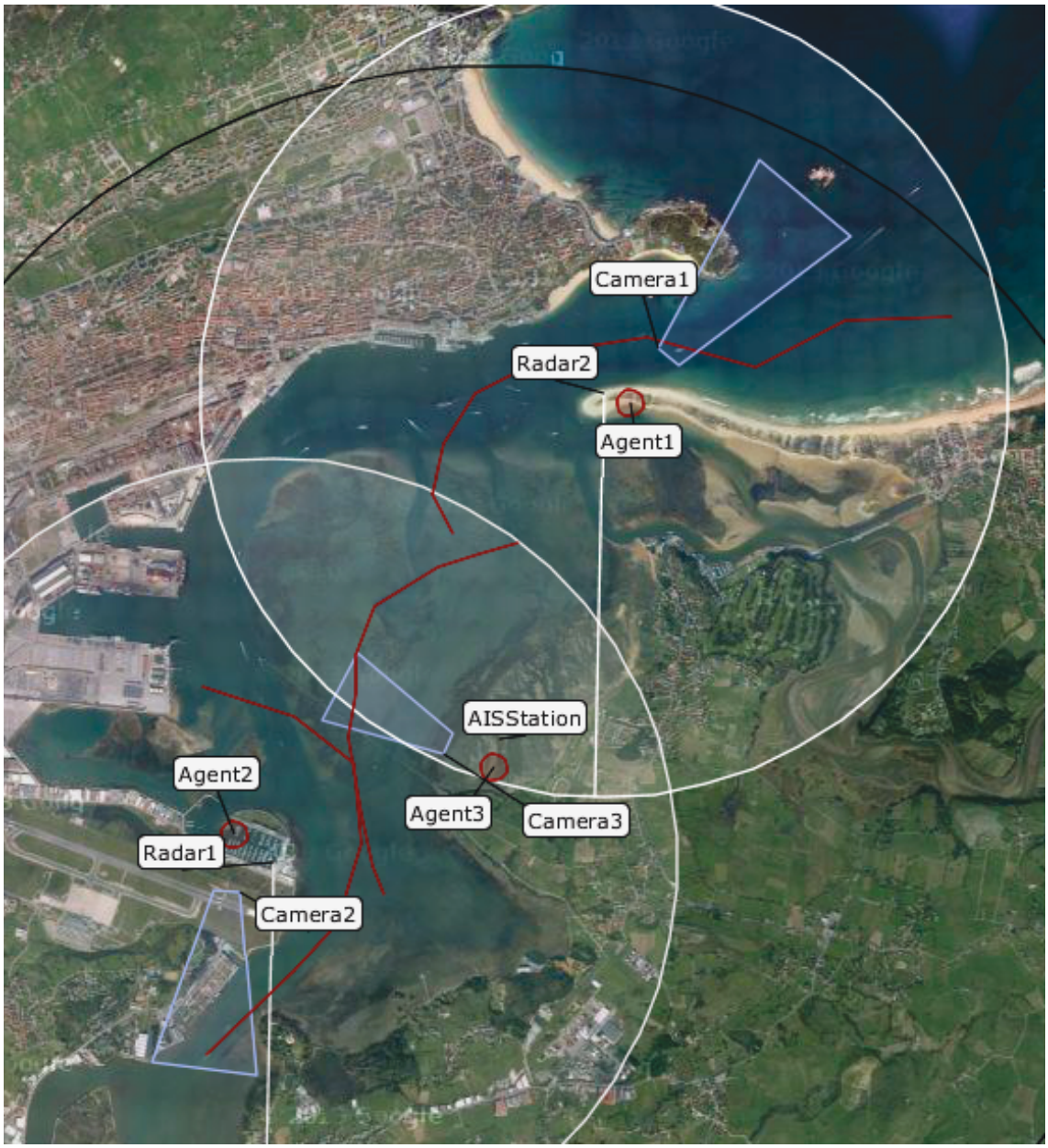


Figura 5.15: Descripción gráfica del entorno y los sensores desplegados. Podemos observar dos radares (cuyo rango se representa en blanco), y una estación AIS (su cobertura aparece en negro). Además hay tres cámara PTZ desplegadas en las posiciones de cada uno de los sensores (se puede ver su campo de visión actual en azul). Cada cámara tiene además definidas unas zonas de monitorización (en rojo).

Se proporciona una descripción más precisa de los diferentes sensores desplegados en la tabla 5.5. Se puede observar que los alances de los radares y la estación AIS no son del todo realistas. Un radar costero, dependiendo de las condiciones de su despliegue podría tener una visibilidad de unos 60Km. Una estación AIS por otro lado podría tener una cobertura de entre unos 30 a 100Km. En este caso se han simulado alcances más pequeños para poder representar correctamente el escenario, ya que además no vamos experimentar con trayectorias simuladas muy lejanas.

| Sensor | Tipo | Ubicación (WGS84) | Descripción |
|---|---|---|---|
| Radar1 | Radar | $43{,}4272°, -3{,}8060°$ | Alcance 3000m. Período 3s. |
| Radar2 | Radar | $43{,}4585°, -3{,}7756°$ | Alcance 3000m. Período 2s. |
| EstaciónAIS | AIS | $43{,}4354°, -3{,}7853°$ | Alcance 5000m. |
| Cámara1 | Cámara PTZ | $43{,}4579°, -3{,}7732°$ | Alcance 3000m. 25 FPS. |
| Cámara2 | Cámara PTZ | $43{,}4291°, -3{,}8096°$ | Alcance 3000m. 25 FPS. |
| Cámara3 | Cámara PTZ | $43{,}4336°, -3{,}7858°$ | Alcance 3000m. 25 FPS |

Tabla 5.5: Descripción de sensores utilizados para la experimentación en el entorno de vigilancia marítimo.

Por último, para la gestión de este entorno ha sido necesario instanciar determinados agentes que representan tanto los sensores, como la lógica de gestión, así como la fusión de datos. Estos agentes, junto con la entrada y salida relacionada puede consultarse en la tabla 5.6. Estos agentes son los descritos en la sección 5.3. En este caso estos agentes han sido simulados también dentro del entorno de simulación, que también permite el despliegue de agentes, simular la comunicación y su procesamiento asíncrono, etc.

Con toda esta información representada dentro del simulador podemos estar en disposición de ejecutar la simulación y recolectar información del entorno a través de los sensores, determinada información sobre el estado de los agentes, los objetivos monitorizados, etc. Estas pruebas se describen en más detalle en la siguiente sección.

| Agente | Tipo | Entrada | Salida |
|---|---|---|---|
| AgenteRadar1 | Agente Sensor | N/A | Pistas Locales Radar1 |
| AgenteRadar2 | Agente Sensor | N/A | Pistas Locales Radar2 |
| AgenteAIS | Agente Sensor | N/A | Pistas Locales AIS |
| AgenteCámara1 | Agente Sensor | Control AgenteControl1 | Pistas Locales Cámara1 |
| AgenteCámara2 | Agente Sensor | Control AgenteControl2 | Pistas Locales Cámara2 |
| AgenteCámara3 | Agente Sensor | Control AgenteControl3 | Pistas Locales Cámara3 |
| AgenteFusion | Agente Fusión | Pistas Locales: AgenteRadar1, AgenteRadar2, AgenteRadar3, AgenteAIS | Pistas Globales Fusión |
| AgenteControl1 | Agente Control | Pistas Locales o Globales dependiendo del experimento | Control Cámara1 |
| AgenteControl2 | Agente Control | Pistas Locales o Globales dependiendo del experimento | Control Cámara2 |
| AgenteControl3 | Agente Control | Pistas Locales o Globales dependiendo del experimento | Control Cámara3 |

Tabla 5.6: Descripción de los agentes instanciados para la experimentación en el entorno de vigilancia marítimo.

### 5.5.2 Descripción de las pruebas

El proceso de evaluación consistirá en la definición de un conjunto de embarcaciones y sus trayectorias. Estas embarcaciones en movimiento serán susceptibles de ser monitorizadas por las cámaras PTZ dependiendo de su ubicación en el entorno. El sistema multi-agente percibirá la información generada por los sensores en tiempo real, y decidirá la mejor embarcación a ser monitorizada en cada momento, aplicando para ello los algoritmos de priorización y coordinación descritos en la sección 5.3.3. Se aplicarán estos algoritmos utilizando diferentes fuentes de información: la generada únicamente por las cámaras, los sensores, o la fusión de información. Esto nos permitirá evaluar el comportamiento del sistema multi-agente cuando se recibe información más o menos completa.

El uso de esta información por parte de los Agentes de Control (*CA* se especifica a continuación:

*CA* **Usando información de las Cámaras**

En este caso los Agentes de Control únicamente percibirán información del entorno a través de las cámaras. De este modo, un agente de control sólo podrá comenzar a monitorizar una embarcación si esta ha pasado por el campo de visión de la cámara.

*CA* **Usando información de las Cámaras y Sensores**

En este segundo experimento los Agentes de Control no sólo recibirán información de las cámaras, si no también de los sensores desplegados en el entorno. En este caso los agentes no requieren estar mirando a alguna embarcación para saber que está presente, ya que esta será detectada y reportada por los sensores que cubran su localización.

*CA* **Usando Información de Fusión**

En este último experimento los Agentes de Control recibirán la información de un Agente de Fusión que estará fusionando la información de las diferentes cámaras y sensores.

Por otra parte, las métricas que se evaluarán para comprobar el rendimiento del sistema multi-agente en el control automático de las cámaras quedarán determinadas por los siguientes elementos:

**Tiempo de escaneo (1)**

Representa el tiempo total en que los Agentes de Control han estado monitorizando el entorno en busca de nuevos objetivos o simplemente monitorizando las áreas de interés que tienen definidas.

**Tiempo de monitorización (2)**

Indica el tiempo total en que los Agentes Control han estado monitorizando embarcaciones del entorno.

**Tiempo efectivo de monitorización (3)**

Indica el tiempo total en que los Agentes Control han estado monitorizando embarcaciones distintas del entorno (monitorización no redundante). Cuanto más alto sea este valor, mejor se estará comportando el sistema multi-agente.

**Tiempo redundante de monitorización (4)**

Indica el tiempo total en que los Agentes Control han estado monitorizando las mismas embarcaciones del entorno (monitorización redundante). El valor óptimo para este indicador debería ser cercano a 0. Si este valor no es 0, es porque ha habido agentes que han estado malgastando el tiempo en la monitorización redundante mientras que podrían haber estado monitorizando otras embarcaciones (si las hubiera), o el entorno.

**Tiempo perdido (5)**

Este indicador se refiere al tiempo total en el que los agentes han empleado su tiempo en

la monitorización del entorno, o realizando una monitorización redundante, cuando había disponible otras embarcaciones que no estaban siendo monitorizadas. Cuanto menor sea este valor, mejor será el desempeño del sistema multi-agente.

Estos valores se obtienen directamente del proceso de simulación, donde para cada instante de simulación se almacena el estado actual de cada agente, y el estado actual del entorno, para así calcular las métricas que nos permitirán evaluar el rendimiento. Por cada escenario diseñado se realizarán tres ejecuciones de simulación: Uno con Agentes de Control que utilizan únicamente la información de las cámaras, otra con la información de los sensores, y la última con la información de fusión. El resultado de estas ejecuciones sobre cada escenario se describe a continuación.

### 5.5.3 Evaluación de monitorización redundante

El primer experimento consiste en la creación de una única embarcación a ser monitorizada que describe una trayectoria que pasa por el área de cobertura de todos los sensores y cámaras. En este caso, como contamos con más cámaras que embarcaciones, es posible que varias cámaras puedan decidir monitorizar el mismo objetivo en vez de vigilar otras áreas del entorno. La idea es que sólo una cámara a la vez pueda estar monitorizando a un blanco determinado, por lo que en este experimento podremos evaluar la necesidad de la fusión de información para satisfacer este requisito.

La trayectoria descrita por la única embarcación comienza en la posición $43{,}4357°, -3{,}8029°$ (cercana al Radar1 de la figura 5.15) y termina en $43{,}4709°, -3{,}7580°$ (cerca del Radar2 en la figura 5.15), y se mueve a una velocidad constante de $10m/s$ con ligeras maniobras de $2°/s$. La duración total de esta trayectoria ronda los 580 segundos y puede ser cubierta en su totalidad por las cámaras del entorno.

A continuación se describen los resultados de los experimentos que se han realizado sobre este entorno.

#### 5.5.3.1 Usando cámaras

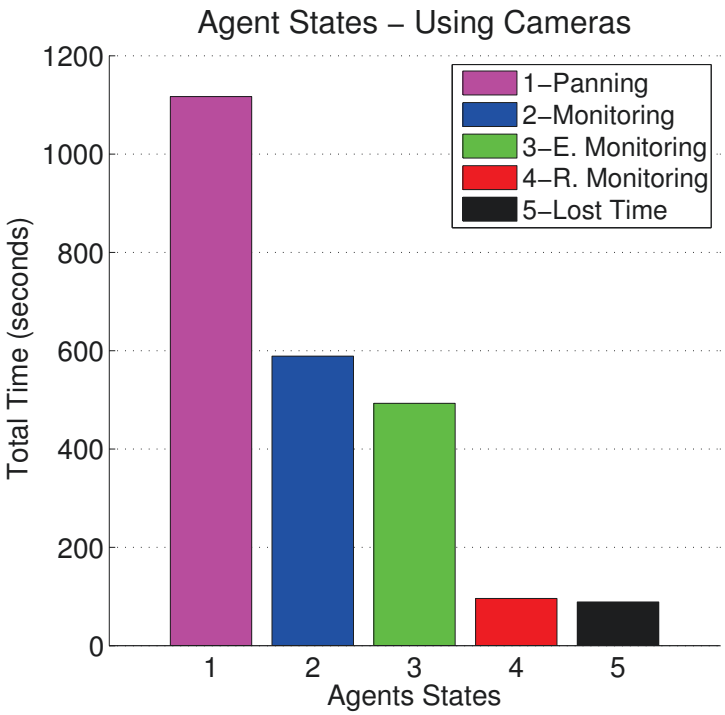


Figura 5.16: Resultado de evaluación del sistema Multi-Agente con una única pista a monitorizar cuando utilizamos las cámaras como única fuente de información.

En este caso los Agentes de Control cuentan únicamente con la información proporcionada por las cámaras para realizar las tareas de monitorización. En la figura 5.16 se puede observar que cuando se nos presenta este caso puede ocurrir que haya períodos de tiempo donde no se ha estado monitorizando la única embarcación del entorno (se puede ver en el tiempo perdido). En este caso, como la detección del blanco depende directamente del campo de visión de la cámara, si hay una cámara que no se encuentra visualizando los alrededores de la embarcación, esta pasará desapercibida. También aparece un pequeño período de monitorización redundante que indica que los agentes han estado monitorizando la misma embarcación. Esto es debido a que al no haber información de fusión, cada uno de los agentes percibe el blanco como una entidad diferente. Lo que no dispara los procesos de coordinación pertinentes.

#### 5.5.3.2 Usando cámaras y sensores

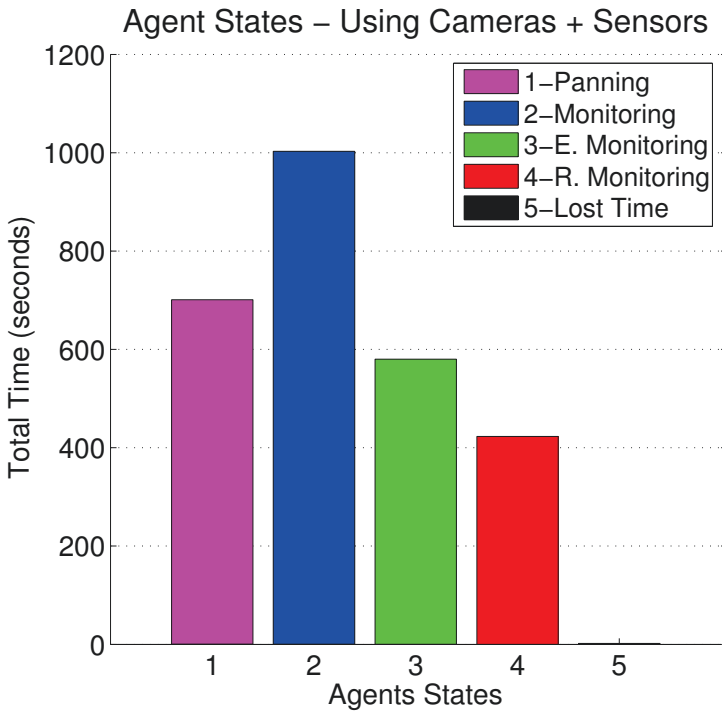


Figura 5.17: Resultado de evaluación del sistema Multi-Agente con una única pista a monitorizar cuando utilizamos las cámaras y sensores como únicas fuentes de información.

En la figura 5.17 se presenta el resultado de la simulación cuando los agentes, además de las cámaras, cuentan con la información proporcionada por el resto de sensores (radar y AIS). Como se puede ver, en esta simulación el tiempo perdido por los agentes ha bajado a cero, ya que en cuanto la embarcación entra en la cobertura de uno de los sensores, este comienza a estar disponible para su monitorización. Independientemente de la posición de la cámara. Por lo tanto, los agentes al percibir las detecciones generadas por los sensores, ajustan la cámara para comenzar la monitorización de la embarcación. Se puede observar, sin embargo, que el tiempo de monitorización redundante ha incrementado lo mismo que ha decrementado el tiempo de escaneo del entorno. Los agentes han pasado de escanear el entorno en busca de objetivos, a monitorizar de manera redundante la embarcación gracias a las detecciones de los sensores. Como los agentes aún no disponen de la información de fusión, cada uno habrá estado siguiendo una pista diferente, como la generada por la cámara, un radar, o un AIS.

#### 5.5.3.3 Usando información de fusión

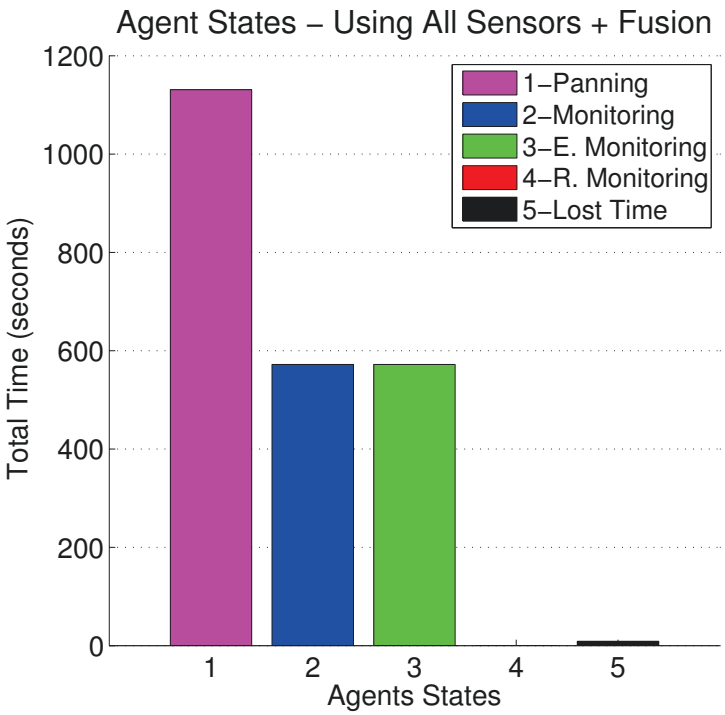


Figura 5.18: Resultado de evaluación del sistema Multi-Agente con una única pista a monitorizar cuando utilizamos la fusión de datos como fuente de información.

Finalmente, en el experimento donde se incluye la fusión como fuente de información de los Agentes de Control, ya podemos observar un comportamiento más razonable. Es de esperar que el tiempo perdido por los agentes se encuentre en los niveles del experimento donde se usan los sensores, pero además que el tiempo de monitorización redundante sea cero. Ya que los agentes podrán coordinarse para evitar monitorizar de la única embarcación del entorno. Este resultado se puede observar en la figura 5.18. El tiempo de monitorización redundante ha bajado a cero, lo que ha permitido a los agentes emplear este tipo en el escaneo del entorno.

Además, podemos ver en la figura 5.19 el estado de cada Agente de Control según avanza la simulación del experimento. La embarcación, al pasar a lo largo del escenario, tiene la posibilidad de ser monitorizada por las diferentes cámaras. Recordemos que uno de los criterios más importantes para monitorizar una embarcación es la distancia que tiene con la cámara. Según avanza la trayectoria por el entorno, la distancia con cada cámara va variando, lo que fuerza la coordinación de los agentes para alternarse en la monitorización. Esto se refleja en la figura, donde se puede observar los períodos que ha empleado cada agente en la monitorización de la pista. Se ve claramente la transición de un agente a otro, y cómo cada agente cambia de tarea cuando otro agente se hace cargo de la monitorización.

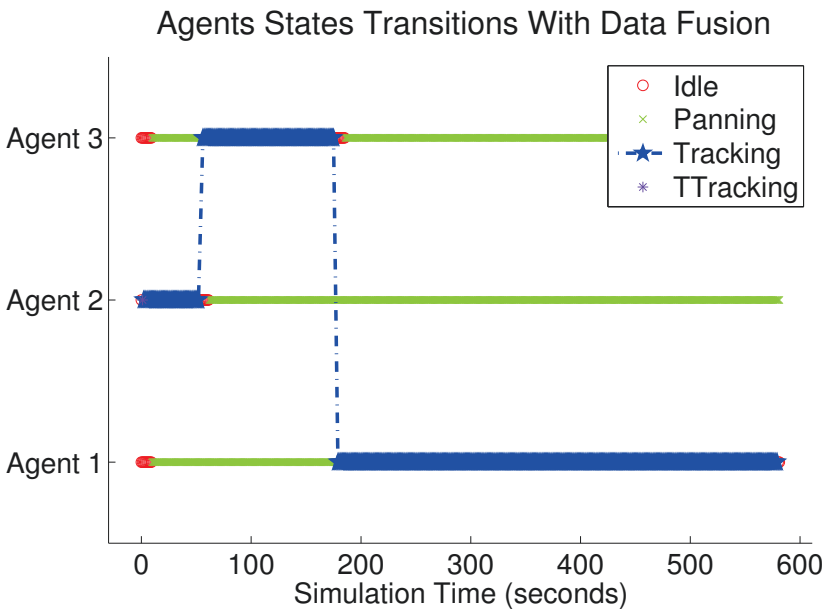


Figura 5.19: Estado de la monitorización de cada uno de los agentes cuando se usa la fusión de datos como fuente de información.

La fusión de datos, es por tanto un aliado indispensable para mejorar los procesos de toma de decisiones en este tipo de entornos. Queda justificada su utilidad y la necesidad de su inclusión dentro de la arquitectura multi-agente.

### 5.5.4 Evaluación de monitorización

En este experimento contaremos con más embarcaciones desplazándose por el entorno. En concreto contamos con tres blancos que describen diferentes trayectorias con una dinámica similar a la del anterior experimento. Esto nos permitirá evaluar el comportamiento del sistema multi-agente cuando contamos con múltiples blancos a seguir, y donde en teoría no debería aparecer monitorización redundante, ya que cada cámara podría llegar a seleccionar un objetivo diferente.

Será interesante comprobar el tiempo de monitorización efectivo en cada simulación, y cómo debería incrementarse con el uso de la fusión de información. Además, el tiempo de escaneo del entorno debería disminuir de igual modo, debido a que los agentes podrán estar ocupados monitorizando las embarcaciones. En las siguientes secciones se exponen los resultados de esta experimentación usando las diferentes fuentes de información, como las cámaras, los sensores, y la fusión.

#### 5.5.4.1 Usando cámaras

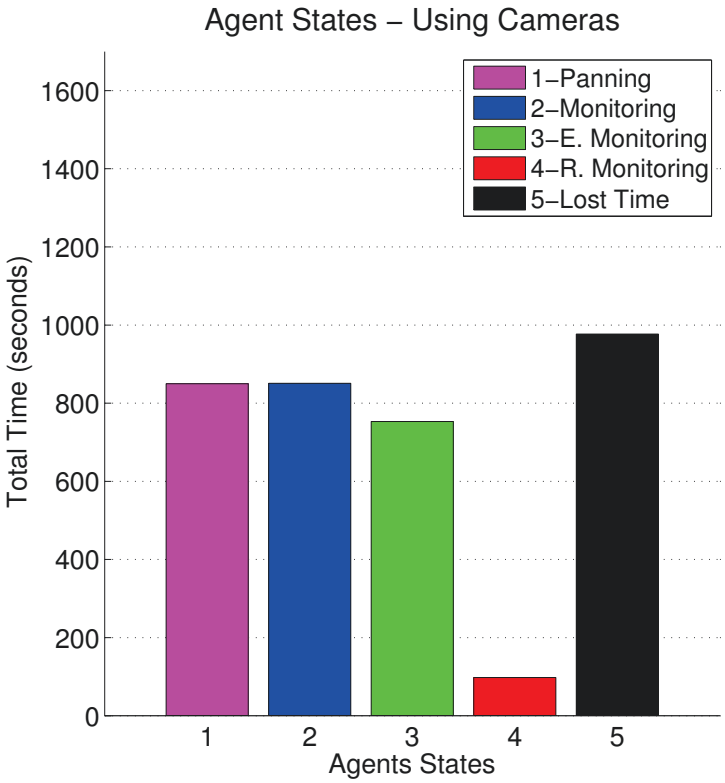


Figura 5.20: Resultado de evaluación del sistema Multi-Agente con múltiples pistas a monitorizar cuando utilizamos las cámaras como única fuente de información.

En este experimento podemos comprobar que con el simple uso de las cámaras, el tiempo perdido por los agentes en la monitorización incrementa de forma notable con respecto al experimento anterior, tal y como se puede observar en la figura 5.20. Esto es debido a que hay más embarcaciones en el entorno, y los Agentes de Control no pueden encontrarlas fácilmente escaneando únicamente el entorno, como se refleja en el tiempo empleado en el escaneo. Esto hace que se dispare el tiempo que han perdido. El tiempo de monitorización redundante más o menos se queda en los mismos niveles que en el experimento anterior.

De nuevo se confirma que utilizando únicamente información proporcionada por las cámaras no es posible realizar una monitorización eficiente de las embarcaciones del entorno.

#### 5.5.4.2 Usando cámaras y sensores

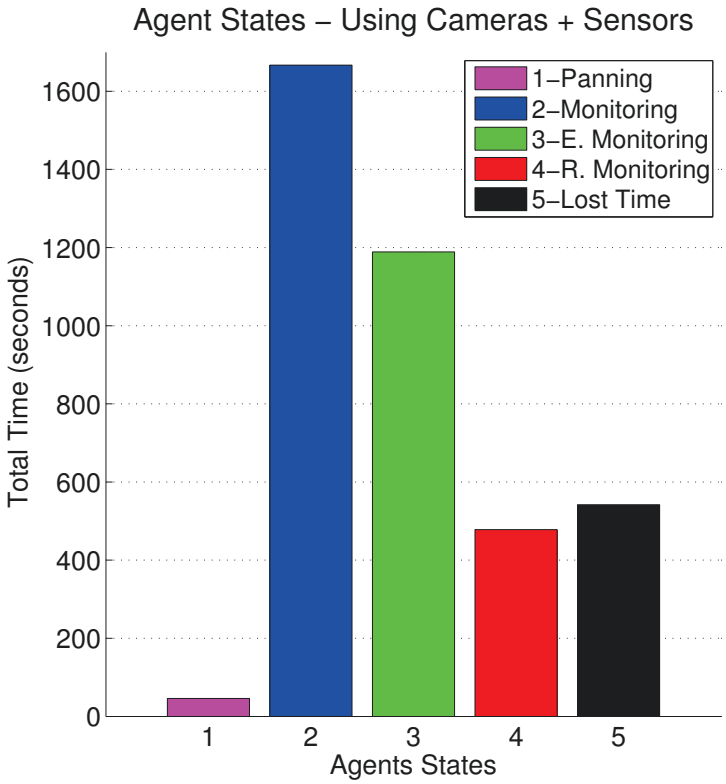


Figura 5.21: Resultado de evaluación del sistema Multi-Agente con múltiples pistas a monitorizar cuando utilizamos las cámaras y sensores como únicas fuentes de información.

Usando además la información de los sensores, se puede ver una reducción notable del tiempo de escaneado en la figura 5.21 en favor del incremento del tiempo de monitorización. El tiempo de monitorización efectiva es menor al tiempo total de monitorización, por lo que en este escenario también se han dado casos de monitorización redundante. Esto se puede observar de nuevo en el tiempo perdido por los agentes, ya que el tiempo de monitorización redundante podría haberse usado para monitorizar el resto de embarcaciones del entorno. Esto no ocurría en el experimento anterior con los sensores ya que no había alternativas de monitorización.

Podemos confirmar por tanto, que los sensores por sí solos ayudan a mejorar la monitorización de las embarcaciones porque mejoran la detección. Los agentes son capaces de aprovechar esta información, pero resulta ser insuficiente para realizar una correcta coordinación.

#### 5.5.4.3 Usando información de fusión

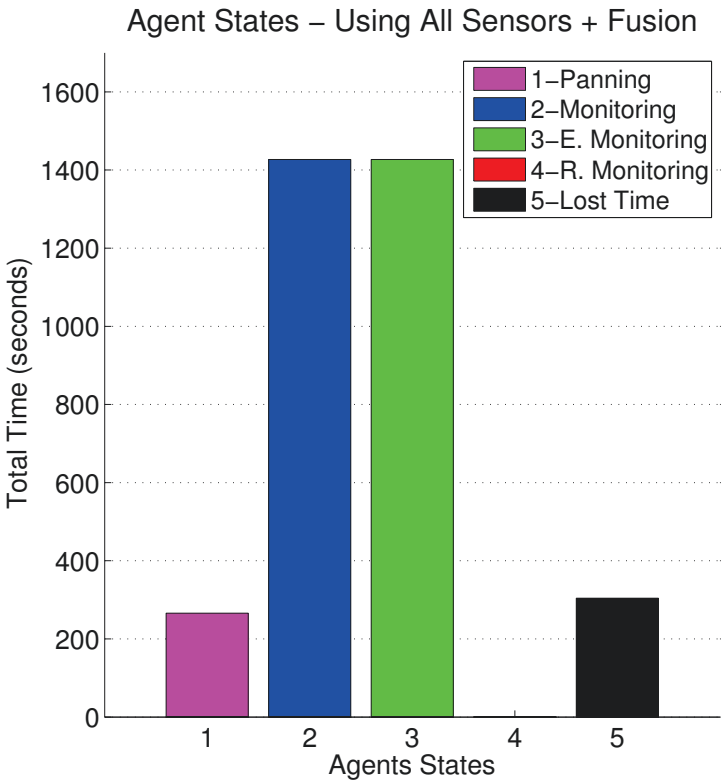


Figura 5.22: Resultado de evaluación del sistema Multi-Agente con múltiples pistas a monitorizar cuando utilizamos la fusión de datos como fuente de información.

Finalmente, el uso de la fusión de datos toma una clara ventaja frente al uso de la información directamente proporcionada por los sensores. Se puede ver en la figura 5.22 cómo el tiempo efectivo en la monitorización se ha igualado al tiempo de monitorización, y que no hay tiempo de monitorización redundante. Sin embargo, en este caso se puede apreciar también que hay cierto tiempo desaprovechado que ha sido empleado en el escaneo del entorno en vez de seguir a las embarcaciones. Esto se debe principalmente a los algoritmos de priorización y coordinación de objetivos descritos en la sección 5.3.3.2.

Por ejemplo, en un momento determinado de la simulación, un Agente de Control $CA_1$ podría monitorizar los blancos $b_1$ y $b_3$. Mientras que el $CA_2$ sólo podría monitorizar $b_1$, ya que $b_3$ queda fuera del alcance. En el cálculo de las prioridades, tanto $CA_1$ como $CA_2$ deciden que el blanco más prioritario es $b_1$, pero es $CA_1$ el que gana la apuesta (porque el blanco se encuentra más cerca de su cámara) y se queda con la monitorización. Automáticamente, el agente $CA_2$ se queda sin poder monitorizar ningún blanco ya que no tiene ninguno objetivo más a su alcance. Esto se podría haber solucionado incluyendo información adicional en el proceso de la puja, de tal forma que el agente $CA_1$ pudiera seleccionar un objetivo subóptimo.

De este modo ambos agentes $CA_1$ y $CA_2$ podrían tener un objetivo que monitorizar, lo que mejoraría el rendimiento global del objetivo.

Por otra parte, podemos observar en las figuras 5.23, 5.24, y 5.25, un mapa de calor que representa las zonas del entorno que han sido cubiertas por las cámaras durante la ejecución de la simulación. Se puede ver claramente cómo el área del entorno que ha sido cubierto con el uso del sistema de fusión es notablemente superior al área cubierto usando simplemente las cámaras o los sensores. Además es interesante comprobar en la figura 5.24, cómo la gran cantidad de monitorización redundante de las embarcaciones ha prevenido la monitorización de otras áreas del entorno, incluso con respecto al uso de cámaras 5.23.

Por lo que este experimento hemos podido comprobar dos hechos diferentes. El primero es que la fusión de datos tiene suma importancia en los entornos donde es necesario tomar decisiones y razonar sobre los diferentes elementos percibidos en un entorno. Y el segundo, es que con la experimentación del entorno simulado hemos podido detectar una posible fuente de mejora en los algoritmos de coordinación entre agentes. Esto nos permitiría rediseñar el mecanismo de coordinación, y repetir el experimento con las mismas condiciones para comprobar la mejorar de la gestión. Algo que sería difícilmente reproducible en un entorno real.

Figura 5.23: Mapa de calor generado por los agentes que han usado sólo las cámaras en su coordinación.

Figura 5.24: Mapa de calor generado por los agentes que han usado sólo las cámaras y los sensores para su coordinación.

Figura 5.25: Mapa de calor generado por los agentes que han usado la fusión de datos para su coordinación.

## 5.6 Conclusiones

En esta aplicación se ha trabajado sobre un entorno de vigilancia marítimo muy similar a entornos reales con los que hemos podido trabajar a lo largo de diferentes proyectos con empresas. Este entorno tiene la particularidad de que cuenta con diferentes tipos de sensores de localización, como los sensores radar, las estaciones AIS, y las cámaras de vigilancia PTZ.

En este tipo de entornos se pone de manifiesto la necesidad de integrar sistemas de gestión y coordinación que permitan ayudar al operador en sus tareas de monitorización. Ya que son sistemas que se encuentran cubriendo grandes áreas del entorno, que normalmente pueden contener un gran número de entidades, áreas de interés a observar, etc. Además estos sistemas cuentan con múltiples sensores que debido a su naturaleza y condiciones de instalación hacen totalmente necesaria la integración de un sistema de fusión que ayude en las tareas de análisis y toma de decisiones. Tanto para el operador al mando del sistema de vigilancia, como para los procesos autónomos que podamos integrar.

En esta sección, por tanto, hemos utilizado el diseño del sistema multi-agente general descrito en la sección 3, y lo hemos aplicado al entorno específico de vigilancia marítima. Para ello se han identificado los agentes necesarios para cubrir este tipo de entornos, y se han descrito las particularidades a implementar en cada uno de ellos. Se ha podido comprobar la utilidad de utilizar un diseño genérico como el propuesto en esta tesis, ya que el diseño contemplado en el entorno multi-cámara de la sección 4 ha resultado ser muy similar al descrito para el entorno marítimo. Esto permitiría reutilizar los componentes que se desarrollen bajo el mismo diseño de base, simplificando el diseño y desarrollo en futuros entornos de gestión multi-sensor.

Para evaluar el sistema multi-agente específico del entorno marítimo se ha diseñado una herramienta de simulación marítima. Esta herramienta, que surgió para cubrir algunos elementos de simulación de trayectorias para algún proyecto con empresa, ha sido extendido con múltiples características que nos han permitido simular al completo este tipo de entornos. Hemos podido simular las embarcaciones, los sensores que realizan las detecciones en función de las áreas de cobertura, cámaras PTZ y su control en tiempo real, y por supuesto la integración agentes de tipo Sensor, Control y Fusión, que representan parte del sistema multi-agente diseñado. Dada la utilidad de esta herramienta de simulación, y el posible interés que podría tener para otros proyectos, estamos trabajando en su documentación y publicación en repositorios de código fuente abierto.

Los experimentos realizados sobre este entorno de vigilancia han permitido evaluar la adecuación del sistema multi-agente diseñado. Se ha podido constatar cómo el sistema diseñado permite ser utilizado con éxito en este tipo de entornos, y cómo además la fusión de datos se vuelve totalmente necesaria para mejorar la toma de decisiones del sistema multi-agente. Cabe remarcar que esta aplicación sobre el entorno marítimo es un ejercicio que nos ha permitido evaluar el diseño del sistema multi-agente en un entorno específico con múltiples sensores

heterogéneos. El diseño, y particularmente los métodos de priorización y coordinación, aún están abiertos a múltiples mejoras y optimizaciones, como hemos podido ver en el transcurso de los experimentos.

# 6

# Conclusiones y Futuros Trabajos

## 6.1 Conclusiones y aportaciones principales

Durante el desarrollo de esta tesis se ha planteado el diseño de un sistema multi-agente para la gestión y coordinación en entornos multi-sensor. Este diseño se ha realizado en base a la definición de los diferentes agentes contemplados, su relación, la información que se transmiten, además del rol que desempeñaría cada agente dentro del entorno multi-sensor. Para este diseño ha sido necesario incluir la fusión de datos como parte fundamental de la arquitectura. Este tipo de sistemas ha demostrado ser de vital importancia para mejorar el análisis y la toma de decisiones dentro del sistema multi-agente, igual que ocurre con los operadores humanos cuando visualizan la salida de la fusión de información.

Este desarrollo ha contemplado el diseño de múltiples agentes, como el Agente Sensor, que representaría la interacción de la arquitectura multi-agente con el entorno. El Agente Fusión ha sido necesario para permitir la integración de la información proporcionada por los Agentes Sensor, y permitir la coordinación y toma de decisiones de los Agentes Control, que son los encargados de realizar la gestión autónoma. Además se han presentado diferentes agentes de interés para el operador y el proceso de integración en un sistema de monitorización real. Estos agentes son los Agentes de Eventos, Agente Operador, y Agente de Interfaz.

Este diseño ha sido evaluado en dos entornos diferentes. Dos entornos que han requerido la especialización del modelo general al entorno particular con el que iba a ser integrado. Estas dos aplicaciones han permitido evaluar la arquitectura multi-agente en situaciones que pueden resultar de gran utilidad en los sistemas de vigilancia actuales. Si bien estas aplicaciones resultan adecuadas para evaluar el proceso de integración del sistema multi-agente en entornos reales, no deben verse como un diseño de referencia, al menos para los procesos de control y coordinación. Ya que estas se han especificado a modo de ilustración sobre cómo integrar una serie de mecanismos autónomos dentro de la arquitectura multi-agente.

La primera aplicación con la que se ha trabajado ha sido con un entorno monitorizado por un conjunto de sensores de visión. Estos sensores son cámaras PTZ que pueden controlarse para monitorizar las diferentes partes o elementos de interés del entorno. En esta aplicación se ha buscado una gestión autónoma para realizar la coordinación en la monitorización de los objetivos del entorno, que en este caso resultaron ser objetos con un determinado color. Las pruebas realizadas en un entorno real, que cuentan con las cámaras distribuidas en diferentes servidores, ha permitido evaluar la adecuación del diseño multi-agente a un entorno tan complejo que requiere una gestión en tiempo real. El resultado de este experimento ha resultado ser satisfactorio, y hemos podido observar cómo un conjunto de agentes distribuidos se coordinaban para monitorizar los diferentes objetivos que aparecen en el entorno.

Este entorno además ha permitido trabajar con sensores físicos reales, como las cámaras PTZ, que han supuesto un reto de integración dentro de la arquitectura multi-agente. Principalmente por la cantidad de información que son capaces de proporcionar, y además de que requieren lógica de control para realizar tareas de vigilancia más eficiente. En este sentido, se ha realizado un amplio trabajo que describe el proceso de integración y optimización de este tipo de sensores en el sistema multi-agente. Este estudio resultó en varias publicaciones en revistas internacionales, así como la publicación de una patente.

La segunda aplicación sobre la que se ha trabajado ha consistido en la integración del sistema multi-agente en un entorno de vigilancia marítimo. Este tipo de entornos cuenta con más variedad de sensores que el entorno multi-cámara, como son los sensores radar y las estaciones AIS. El objetivo de gestión también ha consistido en la coordinación de las diferentes cámaras PTZ para monitorizar las embarcaciones del entorno. Para ello, el sistema multi-agente ha utilizado la información de fusión, que le permite realizar una coordinación más eficiente, tal y como hemos podido observar en la experimentación. Los mecanismos de priorización y optimización se han basado en este caso en la aplicación del algoritmo de análisis jerárquico (AHP), que permite a un eventual operador ajustar correctamente los pesos que describen sus prioridades acerca de las embarcaciones a monitorizar.

A diferencia del entorno multi-cámara, en este entorno marítimo hemos desarrollado un simulador de entornos marítimos que nos han permitido desplegar diferentes entidades como sensores, embarcaciones, agentes, e integrar procesos de medición y evaluación. Este desarrollo ha surgido de la necesidad de probar el diseño multi-agente en este tipo de entornos. Ya de por sí es difícil acceder a los datos que generados por los diferentes sensores, por lo que la integración real en estos sistemas se puede volver aún más complicado. Además, estos sistemas suelen encontrarse en zonas costeras, por lo que el desarrollo e integración en un entorno real sería muy costoso. Sin embargo, la herramienta de simulación nos ha facilitado la integración del diseño y su experimentación, lo que nos ha permitido valorarla muy positivamente. Además hemos podido constatar la necesidad de un sistema de fusión para mejorar las tareas de coordinación entre los agentes. Actualmente estamos en proceso de publicación de la herramienta, así como

de su liberación bajo licencias Open Source.

De este modo podemos concluir que el diseño del sistema multi-agente propuesto ha podido ser integrado con éxito en dos entornos multi-sensor que requieren de una gestión y coordinación en tiempo real. Esto nos ha permitido evaluar la adecuación del diseño general a entornos concretos, así como experimentar sobre entornes reales que siempre cuentan con particularidades que a veces son complicadas de gestionar.

## 6.2 Trabajos Futuros

En esta tesis hemos cubierto determinados aspectos de la gestión y coordinación multi-sensor. Principalmente los relacionados con la identificación de los agentes involucrados y la definición de los flujos de información del sistema. En este entorno además se ha considerado la integración de la fusión de información para solucionar tareas complejas que requieran de colaboración entre agentes. Hemos podido ver dos aplicaciones prácticas que permiten evaluar el diseño, así como servir de referencia para posibles implementaciones reales.

Sin embargo, hay varios aspectos que no ha sido tratados en la tesis y que podrían requerir de cierta atención. El primero puede consistir en la evaluación de la arquitectura en entornos más amplios, con mayor número de sensores, y con mayor número de elementos a coordinar. Esto nos permitiría hacernos una idea de la posible escalabilidad de la arquitectura ante diferentes entornos con carga. Esto podría llevarse a cabo a través de entornos reales, o entornos simulados que tengan la capacidad de distribuirse para simular condiciones reales de funcionamiento. En este tipo de entornos sería muy útil además evaluar diferentes arquitecturas de fusión, ya que en esta tesis, debido al limitado número de sensores de los entornos probados, siempre se ha optado por una arquitectura de fusión centralizada.

Otro punto interesante sería el diseño de mecanismos de auto colaboración entre agentes. En esta tesis hemos visto como los agentes desplegados en cada arquitectura están vinculados los unos con otros. Esto actualmente se estaría realizando de forma estática en la instanciación de cada agente. Pero sería más útil poder desplegar únicamente los Agente Sensor de la arquitectura, y que el resto de agentes necesarios se auto instancien y auto configuren para realizar los procesos de coordinación y gestión. De esta forma el proceso de adaptación y despliegue a un entorno particular resultaría más sencillo y llevadero. Especialmente si se trata de un entorno cambiante donde nos podemos encontrar con la adición o eliminación de sensores.

También sería interesante discutir y evaluar diferentes mecanismos de coordinación entre agentes. En los Agentes de Control que hemos diseñado para cada entorno en particular podemos encontrar un ejemplo concreto de aplicación. Uno basado en reglas y otro en la aplicación algoritmos de análisis jerárquicos. Estos se han diseñado a modo de ilustración para ver un ejemplo real de gestión y coordinación. Pero sería muy interesante poder evaluar otros mecanismos basados en lógica borrosa, redes bayesianas, algoritmos de clasificación, etc. Esto nos permitirá tener un conjunto mayor de herramientas y poder evaluar su adecuación a cada uno de los problemas.

El último punto a tratar, ya desde el punto de vista de la posible integración del diseño de esta tesis en entornos reales, consistiría en el desarrollo de las librerías que soporten de manera general el diseño multi-agente. En el ejemplo multi-cámara se realizó una implementación real pero para el entorno multi-cámara. Podría ser útil para otros casos de uso realizar una implementación de referencia que pueda ser particularizada para cada entorno, igual que hemos

hecho a nivel de diseño. Esta arquitectura ya necesitaría seleccionar la plataforma sobre la que se ejecutarían los agentes, y quizás proporcionar algún diseño de ontología general que permita representar las información que podemos encontrar en el entorno. Pero que esta pueda ser particularizada en el diseño e integración del sistema.

# A

# Apéndice

## A.1 Proyectos relacionados

En el transcurso de la tesis hemos podido colaborar en diferentes proyectos de I+D con diferentes empresas. Estos trabajos han estado de alguna forma relacionados con parte del trabajo desarrollado en esta tesis, en especial la aplicación del entorno marítimo y el uso de técnicas de fusión en sistemas multi-sensor. Algunos de estos proyectos se describen brevemente a continuación.

### A.1.1 AIRBUS Military

Se desarrolló un sistema de fusión de información para ser integrado en un sistema FITS (Fully Integrated Tactical System) embarcado abordo de un avión. Este desarrollo obtenía e integraba información de diferentes fuentes de información como la inteligencia electrónica (ELINT), inteligencia de señales (SIGINT), e inteligencia de comunicaciones (COMINT). El sistema de fusión permitía obtener un conjunto de entidades no redundantes sobre todas las fuentes monitorizadas, así como de realizar un proceso de identificación de entidades. También se diseñó un sistema de fusión de información paralelo que permitía la integración de las detecciones de sensores radar y una estación AIS que van abordo en el avión.

### A.1.2 Elecnor Deimos

En este proyecto trabajamos con un sistema de vigilancia marítimo que cuenta con sensores radar y estaciones AIS muy similares a los descritos en la aplicación del entorno marítimo. Para este entorno se desarrolló un sistema de fusión de datos que permitía integrar la información proporcionada en tiempo real por los diferentes sensores. La particularidad de este sistema de fusión es que es altamente configurable, ya que permite definir diferentes regiones del

entorno donde es posible configurar algoritmos como PDA, JPDA, Kalman, IMM, así como sus parámetros.

### A.1.3 Núcleo de Comunicaciones y Control

Este proyecto es muy similar al realizado en Elecnor Deimos. También trató del desarrollo de un sistema de fusión de información en un entorno marítimo con sensores radar y AIS.

### A.1.4 Kongsberg Norcontrol IT

Este proyecto consistió en el análisis de información de sensores radar y estaciones AIS en entornos reales. En la primera etapa se realizó un estudio sobre diferentes deficiencias e inconsistencias encontradas sobre la información percibida del entorno. En una segunda fase se diseñaron algoritmos de fusión de alto nivel que permitiesen utilizar información contextual del entorno para mejorar determinados procesos de fusión de bajo nivel.

## A.2 Publicaciones relacionadas

El desarrollo de esta tesis se ha soportado por múltiples aportaciones a conferencias, publicaciones en revista, e inclusive el desarrollo de una patente. En concreto se han publicado 3 artículos en revistas internacionales que aparecen en el índice JCR, y 12 publicaciones en diversos congresos internacionales. Estos trabajos se indican a continuación.

### A.2.1 Publicaciones en revistas internacionales (JCR)

- **2013** A practical approach for active camera coordination based on a fusion-driven multi-agent system, International Journal of System Science, ISSN 0020-7721, DOI: 10.1080/00207721.2013.795632
- **2011** MIJ2K: Enhanced Video Transmission Based on Conditional Replenishment of JPEG2000 Tiles with Motion Compensation, Journal of Visual Communication and Image Representation, In Press, Accepted Manuscript, Available online 19 February 2011, ISSN 1047-3203, DOI: 10.1016/j.jvcir.2011.02.002.
- **2011** MIJ2K Optimization using Evolutionary Multiobjective Optimization Algorithms, Expert Systems with Applications. In Press, Accepted Manuscript, ISSN 0957-4174, DOI: 10.1016/j.eswa.2011.02.143

### A.2.2 Publicaciones en conferencias

- **2014** Geographic context configuration in fusion algorithms for maritime surveillance". Fusion. 17th International Conference on Information Fusion.
- **2014** Information fusion as input source for improving multi-agent system autonomous decision-making in maritime surveillance scenarios". Fusion. 17th International Conference on Information Fusion.
- **2013** Fusion of sensor data and intelligence in FITS. E. Martí, A. Luis, J. García, S. Onate, C Sánchez, S. Gonzalez. In Information Fusion (FUSION) 16th International Conference on (pp. 342–349). Istanbul: IEEE.
- **2012** Distributed Active-Camera Control Architecture Based on Multi-Agent Systems", A. Luis, J. M. Molina, M. A. Patricio". Highlights on Practical Applications of Agents and Multi-Agent Systems. Advances in Intelligent and Soft Computing, Vol. 156, pp. 103-112. Springer Berlin / Heidelberg. Salamanca, Spain. Marzo 28-30, 2012.
- **2012** ContextCare: Autonomous Video Surveillance System Using Multi-camera and Smartphones", Gonzalo Blázquez Gil, Alvaro Luis Bustamante, Antonio Berlanga, José M. Molina. Management Intelligent Systems. Advances in Intelligent Systems and Computing Volume 171, 2012, pp 47-56. Springer Berlin / Heidelberg.
- **2011** Autonomous active-camera control architecture based on Multi-Agent Systems for surveillance scenarios", A. Luis, J. M. Molina, M. A. Patricio". Prediction and Recognition of Piracy Efforts Using Collaborative Huma-Centric Information Systems. Salamanca 2011.
- **2011** Multi-camera Control and Video Transmission Architecture for Distributed Systems", A. Luis, J. M. Molina, M. A. Patricio". User-Centric Technologies and Applications 2011. Advances in Intelligent and Soft Computing. Salamanca 2011.
- **2010** Multi-Camera and Multi-Modal Sensor Fusion, an Architecture Overview", A. Luis, J. M. Molina, M. A. Patricio". International Symposium on Distributed Computing and Artificial Intelligence 2010 DCAI 2010, Workshop CONTEXTS 2010.Valencia, España, Septiembre 7-10, 2010.
- **2010** Robust Sensor Fusion in Real Maritime Surveillance Scenarios". Jesús García, José Luis Guerrero, Alvaro Luis, José M. Molina. 13th International Conference on Information Fusion. 26-29 July 2010 EICC. Edinburgh, UK.
- **2009** Fast Summarizing of MIJ2K Video Sequences Creating Storyboards". A. Luis, José M. Molina, Miguel A. Patricio. Third International Workshop o User-Centric Technlogies and applications (MADRINET 2009). Salamanca, España.

- **2009** Video Encoder Optimization via Evolutionary Multiobjective Optimization Algorithms". A.Luis, José M. Molina, Miguel A. Patricio. The Genetic and Evolutionary Computation Conference (GECCO 2009). Montreal, Canadá.

- **2008** Scalable Streaming of JPEG 2000 Live Video Using RTP". A. Luis, Miguel A. Patricio. International Symposium on Distributeed Computing and Artificial Intelligence 2008 (DCAI 2008). Salamanca, España.

### A.2.3 Patentes

- **2012** Método de codificación y compresión interframe de vídeo con JPEG 2000. ES 2 368 050 B1

# Bibliografía


Abidi, B. R., Aragam, N. R., Yao, Y., and Abidi, M. A. (2008). Survey and analysis of multimodal sensor planning and integration for wide area surveillance. *ACM Computing Surveys (CSUR)*, 41(1):7.

Adams, M. D. and Ward, R. (2001). Wavelet transforms in the jpeg-2000 standard. In *Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, BC, Canada*, volume 1, pages 160–163. Citeseer.

Ahmed, A. A., Shi, H., and Shang, Y. (2005). Sharp: A new approach to relative localization in wireless sensor networks. In *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pages 892–898. IEEE.

Akselrod, D., Sinha, A., and Kirubarajan, T. (2007). Collaborative distributed sensor management for multitarget tracking using hierarchical markov decision processes. In *Optical Engineering+ Applications*, pages 669912–669912. International Society for Optics and Photonics.

Akyildiz, I. and Kasimoglu, I. (2004). Wireless sensor and actor networks: research challenges. *Ad Hoc Networks*, 2(4):351–367.

Al Haj, M., Bagdanov, A. D., Gonzalez, J., and Roca, F. (2010). Reactive object tracking with a single ptz camera. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 1690–1693. IEEE.

An, B., Sim, K. M., Miao, C. Y., and Shen, Z. Q. (2008). Decision making of negotiation agents using markov chains. *Multiagent and Grid Systems*, 4(1):5–23.

Aoki, E. H., Bagchi, A., Mandal, P., and Boers, Y. (2011). A theoretical look at information-driven sensor management criteria. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1–8. IEEE.

Bar-Shalom, Y. and Tse, E. (1975). Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451–460.

Becker, G. and Güdesen, A. (2000). Passive sensing with acoustics on the battlefield. *Applied Acoustics*, 59(2):149–178.

Bedworth, M. and O'Brien, J. (2000). The omnibus model: a new model of data fusion? *IEEE Aerospace and Electronic Systems Magazine*, 15(4):30–36.

Bellifemine, F., Poggi, A., and Rimassa, G. (2001). Jade: a fipa2000 compliant agent development environment. In *Proceedings of the fifth international conference on Autonomous agents*, pages 216–217. ACM.

Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P., and Bertsekas, D. P. (1995). *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA.

Bevington, J. E. (2004). Distributed sensor management and target tracking for unattended ground sensor networks. In *Defense and Security*, pages 25–35. International Society for Optics and Photonics.

Blackrnan, S. and House, A. (1999). Design and analysis of modern tracking systems. *Boston, MA: Artech House*.

Blom, H. A. and Bar-Shalom, Y. (1988). The interacting multiple model algorithm for systems with markovian switching coefficients. *Automatic Control, IEEE Transactions on*, 33(8):780–783.

Boliek, M. (2000). Information technology jpeg2000 image coding system-part 3: Motion jpeg 2000. *ISO/IEC IS*, pages 15444–1.

Bossé, É., Roy, J., and Wark, S. (2007). *Concepts, models, and tools for information fusion*. Artech House ^ eBoston Boston.

Boussard, M., Bouzid, M., and Mouaddib, A.-I. (2007). Multi-criteria decision making for local coordination in multi-agent systems. In *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, volume 2, pages 87–90. IEEE.

Bramberger, M., Quaritsch, M., Winkler, T., Rinner, B., and Schwabach, H. (2005). Integrating multi-camera tracking into a dynamic task allocation system for smart cameras. In *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*, pages 474–479. IEEE.

Braubach, L., Lamersdorf, W., and Pokahr, A. (2003). Jadex: Implementing a bdi-infrastructure for jade agents.

Broida, T. J. and Chellappa, R. (1986). Estimation of object motion parameters from noisy images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):90–99.

Bustamante, A. and Molina, J. (2011). Multi-camera Control and Video Transmission Architecture for Distributed Systems. *User-Centric Technologies and*, pages 1–9.

Bustamante, A. L., Molina, J. M., and Patricio, M. A. (2013). Autonomous active-camera control architecture based on multi-agent systems for surveillance scenarios. *Prediction and Recognition of Piracy Efforts Using Collaborative Human-Centric Information Systems*, 109:210.

Carthel, C., Coraluppi, S., and Grignan, P. P. (2007). Multisensor tracking and fusion for maritime surveillance. In *FUSION*, pages 1–6.

Castanedo, F., García, J., Patricio, M. A., and Molina, J. M. (2010). Data fusion to improve trajectory tracking in a Cooperative Surveillance Multi-Agent Architecture. *Information Fusion*, 11(3):243–255.

Castanedo, F., Patricio, M. A., Garcia, J., and Molina, J. M. (2006). Extending surveillance systems capabilities using bdi cooperative sensor agents. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 131–138. ACM.

Charlish, A., Woodbridge, K., and Griffiths, H. (2012). Multi-target tracking control using continuous double auction parameter selection. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 1269–1276. IEEE.

Charrier, M., Cruz, D., and Larsson, M. (1999). Jpeg2000, the next millennium compression standard for still images. In *Multimedia Computing and Systems, 1999. IEEE International Conference on*, volume 1, pages 131–132. IEEE.

Chen, K., Luo, X.-S., and Liu, X. (2005). Sensor resource management research based on intelligent agent in naval multi-platform cooperative engagement. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 1, pages 132–136. IEEE.

Chong, E. K., Kreucher, C. M., and Hero Iii, A. O. (2009). Partially observable markov decision process approximations for adaptive sensing. *Discrete Event Dynamic Systems*, 19(3):377–422.

Christopoulos, C., Skodras, A., and Ebrahimi, T. (2000). The jpeg2000 still image coding system: an overview. *Consumer Electronics, IEEE Transactions on*, 46(4):1103–1127.

Cilla Ugarte, R. (2012). Action recognition in visual sensor networks: a data fusion perspective.

Collins, R. T., Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P., et al. (2000). *A system for video surveillance and monitoring*, volume 2. Carnegie Mellon University, the Robotics Institute Pittsburg.

Comaniciu, D. and Ramesh, V. (2000). Robust detection and tracking of human faces with an active camera. In *Visual Surveillance, 2000. Proceedings. Third IEEE International Workshop on*, pages 11–18. IEEE.

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577.

Conaire, C. O., O'Connor, N. E., Cooke, E., and Smeaton, A. F. (2006). Comparison of fusion methods for thermo-visual surveillance tracking. In *FUSION*, pages 1–7.

Cox, I. J. and Hingorani, S. L. (1996). An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(2):138–150.

Daniyal, F. and Cavallaro, A. (2011). Multi-camera scheduling for video production. In *Visual Media Production (CVMP), 2011 Conference for*, pages 11–20. IEEE.

Daniyal, F., Taj, M., and Cavallaro, A. (2010). Content and task-based view selection from multiple video streams. *Multimedia tools and applications*, 46(2-3):235–258.

Das, S. (2008). *High-level data fusion*. Artech House.

Dasarathy, B. V. (1994). *Decision fusion*, volume 1994. IEEE Computer Society Press Los Alamitos.

Dasarathy, B. V. (1997). Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38.

De Maesschalck, R., Jouan-Rimbaud, D., and Massart, D. L. (2000). The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18.

DeLoach, S. A. (1999). Multiagent systems engineering: a methodology and language for designing agent systems. Technical report, DTIC Document.

Deng, Y. and Manjunath, B. (2001). Unsupervised segmentation of color-texture regions in images and video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(8):800–810.

Devaux, F., Meessen, J., Parisot, C., Delaigle, J.-F., Macq, B., and De Vleeschouwer, C. (2007). A flexible video transmission system based on jpeg 2000 conditional replenishment with multiple references. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 1, pages I–825. IEEE.

Dhillon, S. S. and Chakrabarty, K. (2003). *Sensor placement for effective coverage and surveillance in distributed sensor networks*, volume 3. IEEE.

Dockstader, S. L. and Tekalp, A. M. (2001). Multiple camera fusion for multi-object tracking. In *Multi-Object Tracking, 2001. Proceedings. 2001 IEEE Workshop on*, pages 95–102. IEEE.

Dodin, P., Verliac, J., and Nimier, V. (2000). Analysis of the multisensor multitarget tracking resource allocation problem. In *Proceedings, 3rd International Conference on Information Fusion*.

Doyle, D. D., Jennings, A. L., and Black, J. T. (2013). Optical flow background subtraction for real-time ptz camera object tracking. In *Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International*, pages 866–871. IEEE.

Durrant-Whyte, H., Stevens, M., and Nettleton, E. (2001). Data fusion in decentralised sensing networks. In *4th International Conference on Information Fusion*, pages 302–307.

Dutech, A. and Scherrer, B. (2013). Partially observable markov decision processes. *Markov Decision Processes in Artificial Intelligence*, pages 185–228.

Ebrahimi, T. and Pereira, F. (2002). *The MPEG-4 book*. Number LTS-BOOK-2007-001. Prentice Hall PTR.

El-Barkouky, A., Rara, H., Farag, A., and Womble, P. (2012). Face detection at a distance using saliency maps. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 31–36. IEEE.

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):32–64.

Endsley, M. R. et al. (2000). Theoretical underpinnings of situation awareness: A critical review. *Situation awareness analysis and measurement*, pages 3–32.

Esteban, J., Starr, A., Willetts, R., Hannah, P., and Bryanston-Cross, P. (2005). A review of data fusion models and architectures: towards engineering guidelines. *Neural Computing & Applications*, 14(4):273–281.

Evennou, F. and Marx, F. (2006). Advanced integration of wifi and inertial navigation systems for indoor mobile positioning. *Eurasip journal on applied signal processing*, 2006:164–164.

Everts, I., Sebe, N., and Jones, G. (2007). Cooperative object tracking with multiple ptz cameras. In *Proceedings of the 14th International Conference on Image Analysis and Processing*, pages 323–330. Modena, Italy.

Farmani, N., Sun, L., and Pack, D. (2014). Optimal uav sensor management and path planning for tracking multiple mobile targets. In *ASME 2014 Dynamic Systems and Control Conference*, pages V002T25A003–V002T25A003. American Society of Mechanical Engineers.

Foo, P. H. and Ng, G. W. (2013). High-level information fusion: An overview. *J. Adv. Inf. Fusion*, 8(1):33–72.

Foresti, G. L., Micheloni, C., Snidaro, L., Remagnino, P., and Ellis, T. (2005). Active video-based surveillance system: the low-level image and video processing techniques needed for implementation. *Signal Processing Magazine, IEEE*, 22(2):25–37.

Forsyth, D. A. and Ponce, J. (2002). *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference.

Fortmann, T. E., Bar-Shalom, Y., and Scheffe, M. (1983). Sonar tracking of multiple targets using joint probabilistic data association. *Oceanic Engineering, IEEE Journal of*, 8(3):173–184.

Fossel, S., Fottinger, G., and Mohr, J. (2003). Motion jpeg2000 for high quality video systems. *Consumer Electronics, IEEE Transactions on*, 49(4):787–791.

Fu, Y., Ling, Q., and Tian, Z. (2012). Distributed sensor allocation for multi-target tracking in wireless sensor networks. *Aerospace and Electronic Systems, IEEE Transactions on*, 48(4):3538–3553.

Fukuhara, T., Katoh, K., Kimura, S., Hosaka, K., and Leung, A. (2000). Motion-jpeg2000 standardization and target market. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 2, pages 57–60. IEEE.

Futemma, S., Itakura, E., and Leung, A. (2008). RTP Payload Format for JPEG 2000 Video Streams. RFC 5371 (Informational).

Gad, A. and Farooq, M. (2002). Data fusion architecture for maritime surveillance. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 1, pages 448–455. IEEE.

Galego, R., Bernardino, A., and Gaspar, J. (2012). Auto-calibration of pan-tilt cameras including radial distortion and zoom. In Bebis, G., Boyle, R., Parvin, B., Koracin, D., Fowlkes, C., Wang, S., Choi, M.-H., Mantler, S., Schulze, J., Acevedo, D., Mueller, K., and Papka, M., editors, *Advances in Visual Computing*, volume 7431 of *Lecture Notes in Computer Science*, pages 169–178. Springer Berlin / Heidelberg.

Gans, N., Hu, G., and Dixon, W. (2009). Keeping multiple objects in the field of view of a single ptz camera. In *American Control Conference, 2009. ACC'09.*, pages 5259–5264. IEEE.

García, J., Carbó, J., and Molina, J. M. (2005). Agent-based coordination of cameras. *IJCSA*, 2(1):33–37.

García, J., Guerrero Madrid, J. L., Luis Bustamante, Á., and Molina, J. M. (2010). Robust sensor fusion in real maritime surveillance scenarios.

Georgeff, M., Pell, B., Pollack, M., Tambe, M., and Wooldridge, M. (1999). The belief-desire-intention model of agency. *Intelligent Agents V: Agents Theories, Architectures, and Languages*, pages 1–10.

Giraud, C. and Jouvencel, B. (1994). Sensor selection in a fusion process: a fuzzy approach. In *Multisensor Fusion and Integration for Intelligent Systems, 1994. IEEE International Conference on MFI'94.*, pages 599–606. IEEE.

Girod, B. (1993). What's wrong with mean-squared error? In *Digital images and human vision*, pages 207–220. MIT press.

Gonsalves, P. G. and Rinkus, G. J. (1998). Intelligent fusion and asset management processor. In *Information Technology Conference, 1998. IEEE*, pages 15–18. IEEE.

Gualdi, G., Prati, A., Cucchiara, R., Ardizzone, E., La Cascia, M., Lo Presti, L., and Morana, M. (2008). Enabling technologies on hybrid camera networks for behavioral analysis of unattended indoor environments and their surroundings. In *Proceedings of the 1st ACM workshop on Vision networks for behavior analysis*, pages 101–108. ACM.

Hall, D., Chong, C.-Y., Llinas, J., and Liggins II, M. (2012). *Distributed data fusion for network-centric operations*. CRC Press.

Hall, D. and Llinas, J. (2001). Multisensor data fusion. *Handbook of Multisensor Data Fusion. CRC Press LLC*.

Hall, D. L. and Llinas, J. (1997). An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23.

Hampapur, A. (2008). Smart video surveillance for proactive security. *IEEE Signal Processing Magazine*, 25(4):136.

Han, J. and Bhanu, B. (2007). Fusion of color and infrared video for moving human detection. *Pattern Recognition*, 40(6):1771–1784.

Haskell, B. G. (1997). *Digital Video: An Introduction to MPEG-2: An Introduction to MPEG-2*. Springer Science & Business Media.

Heikkila, J. (2000). Geometric camera calibration using circular control points. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(10):1066–1077.

Hero, A. O., Castañón, D., Cochran, D., and Kastella, K. (2007). *Foundations and applications of sensor management*. Springer Science & Business Media.

Hintz, K. J. and McVey, E. S. (1991). Multi-process constrained estimation. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(1):237–244.

Hodge, L. and Kamel, M. (2003). An agent-based approach to multisensor coordination. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 33(5):648–661.

Horridge, M., Knublauch, H., Rector, A., Stevens, R., and Wroe, C. (2004). A practical guide to building owl ontologies using the protégé-owl plugin and co-ode tools edition 1.0. *University of Manchester*.

Hossain, A. M., Van, H. N., Jin, Y., and Soh, W.-S. (2007). Indoor localization using multiple wireless technologies. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pages 1–8. IEEE.

Initiative, D. C. et al. (2008). Digital cinema system specification, version 1.2.

Isard, M. and Blake, A. (1998). Condensation—conditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28.

ISO/IEC (2000). 15444-1:2000 information technology jpeg2000 image coding system-part 1: core coding system. Technical report, ISO/IEC.

Iyengar, S. S. and Brooks, R. R. (2012). *Distributed Sensor Networks: Sensor Networking and Applications*. CRC press.

Jain, R. and Nagel, H.-H. (1979). On the analysis of accumulative difference pictures from image sequences of real world scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):206–214.

Jin, G.-y., Lu, X.-y., and Park, M.-S. (2006). An indoor localization mechanism using active rfid tag. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, volume 1, pages 4–pp. IEEE.

Johansson, K., Jöred, K., and Svensson, P. (1997). Submarine tracking using multi-sensor fusion and reactive planning for the positioning of passive sonobuoys. *Hydroakustik*, 97.

Jones, G., Renno, J., and Remagnino, P. (2002). Auto-calibration in multiple-camera surveillance environments. In *Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, June*, volume 1, pages 40–47. Citeseer.

Joshi, S. and Boyd, S. (2009). Sensor selection via convex optimization. *Signal Processing, IEEE Transactions on*, 57(2):451–462.

Kalandros, M. and Pao, L. Y. (1998). Controlling target estimate covariance in centralized multisensor systems. In *American Control Conference, 1998. Proceedings of the 1998*, volume 5, pages 2749–2753. IEEE.

Kalandros, M., Pao, L. Y., and Ho, Y.-C. (1999). Randomization and super-heuristics in choosing sensor sets for target tracking applications. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 2, pages 1803–1808. IEEE.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45.

Karakowski, J. (1998). 'towards visual data fusion. *NSSDF International Open Session*.

Karlsson, G. (1996). Asynchronous transfer of video. *Communications Magazine, IEEE*, 34(8):118–126.

Katsilieris, F. (2015). *Sensor management for surveillance and tracking: An operational perspective*. PhD thesis, TU Delft, Delft University of Technology.

Katsilieris, F., Boers, Y., and Driessen, H. (2012). Optimal search: a practical interpretation of information-driven sensor management. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 439–446. IEEE.

Kim, N., Kim, I.-j., and Kim, H.-g. (2006). Video surveillance using dynamic configuration of mutiple active cameras. In *Image Processing, 2006 IEEE International Conference on*, pages 1761–1764. IEEE.

Kincaid, R. K. and Laba, K. E. (1998). Reactive tabu search and sensor selection in active structural acoustic control problems. *Journal of Heuristics*, 4(3):199–220.

Kreucher, C., Kastella, K., and Hero, A. O. (2003). Multi-target sensor management using alpha-divergence measures. In *Information Processing in Sensor Networks*, pages 209–222. Springer.

Kreucher, C. M., Hero, A. O., Kastella, K. D., and Morelande, M. R. (2007). An information-based approach to sensor management in large dynamic networks. *Proceedings of the IEEE*, 95(5):978–999.

Kristensen, S. (1996). *Sensor planning with Bayesian decision analysis*. PhD thesis, Citeseer.

Kumar, R., Wolenetz, M., Agarwalla, B., Shin, J., Hutto, P., Paul, A., and Ramachandran, U. (2003). Dfuse: a framework for distributed data fusion. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 114–125. ACM.

Lee, S. H., Lee, S., Song, H., and Lee, H. S. (2009). Wireless sensor network design for tactical military applications: remote large-scale environments. In *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pages 1–7. IEEE.

Lesser, V. R. and Corkill, D. D. (1989). The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving networks. In *Readings from the AI magazine*, pages 69–85. American Association for Artificial Intelligence.

Li, Y. and Bhanu, B. (2008). Utility-based dynamic camera assignment and hand-off in a video network. In *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pages 1–9. IEEE.

Li, Y. and Bhanu, B. (2011a). A comparison of techniques for camera selection and hand-off in a video network. In *Distributed Video Sensor Networks*, pages 69–83. Springer.

Li, Y. and Bhanu, B. (2011b). Utility-based camera assignment in a video network: A game theoretic framework. *Sensors Journal, IEEE*, 11(3):676–687.

Liggins II, M., Hall, D., and Llinas, J. (2008). *Handbook of multisensor data fusion: theory and practice*. CRC press.

Lim, S., Elgammal, A., and Davis, L. (2003). Image-based pan-tilt camera control in a multi-camera surveillance environment. In *icme*, number 3, pages 645–648. IEEE.

Lin, J. (1991). Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151.

Lindner, J., Murphy, R. R., and Nitz, E. (1994). Learning the expected utility of sensors and algorithms. In *Multisensor Fusion and Integration for Intelligent Systems, 1994. IEEE International Conference on MFI'94.*, pages 583–590. IEEE.

Ling, Q., Fu, Y., and Tian, Z. (2011). Localized sensor management for multi-target tracking in wireless sensor networks. *Information Fusion*, 12(3):194–201.

Llinas, J., Bowman, C., Rogova, G., Steinberg, A., Waltz, E., and White, F. (2004). Revisiting the jdl data fusion model ii. Technical report, DTIC Document.

Lopez, J. M., Rodriguez, F. J., and Corredera, J. C. (1998). Symbolic processing for coordinated task management in multiradar surveillance networks. In *Proceedings of the International Conference on Information Fusion*, pages 725–732. Citeseer.

Lora, M. (1994-2015). Xiph.org :: Test media. World Wide Web electronic publication.

Luis, A. and Patricio, M. A. (2009). Scalable streaming of jpeg 2000 live video using rtp over udp. In *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)*, pages 574–581. Springer.

Luis Bustamante, A., Molina, J. M., and Patricio, M. A. (2014a). A practical approach for active camera coordination based on a fusion-driven multi-agent system. *International Journal of Systems Science*, 45(4):741–755.

Luis Bustamante, A., Molina, J. M., and Patricio, M. A. (2014b). A practical approach for active camera coordination based on a fusion-driven multi-agent system. *International Journal of Systems Science*, 45(4):741–755.

Luis Bustamante, A., Molina López, J. M., and Patricio, M. A. (2009). Video encoder optimization via evolutionary multiobjective optimization algorithms. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1835–1836. ACM.

Maes, P., Guttman, R. H., and Moukas, A. G. (1999). Agents that buy and sell. *Communications of the ACM*, 42(3):81–ff.

Mahler, R. (2003). Objective functions for bayesian control-theoretic sensor management, 1: Multitarget first-moment approximation. In *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, volume 4, pages 4_1905–4_1923. IEEE.

Manyika, J. and Durrant-Whyte, H. (1994). Data fusion and sensor management: A decentralized information-theoretic approach.

Marpe, D., George, V., Cycon, H. L., and Barthel, K. U. (2004). Performance evaluation of motion-jpeg2000 in comparison with h. 264/avc operated in pure intracoding mode. In *Photonics Technologies for Robotics, Automation, and Manufacturing*, pages 129–137. International Society for Optics and Photonics.

Marti, E., Luis, A., González, B., and García, J. (2014). Geographic context configuration in fusion algorithms for maritime surveillance. In *Information Fusion (FUSION), 2014 17th International Conference on Information Fusion*. IEEE.

Mayott, G., Miller, G., Harrell, J., Hepp, J., and Self, M. (2010). System approach to distributed sensor management. In *SPIE Defense, Security, and Sensing*, pages 77070I–77070I. International Society for Optics and Photonics.

McIntyre, G. A. and Hintz, K. J. (1996). Information theoretic approach to sensor scheduling. In *Aerospace/Defense Sensing and Controls*, pages 304–312. International Society for Optics and Photonics.

Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 25(5).

Meessen, J., Parisot, C., Desurmont, X., and Delaigle, J.-F. (2005). Scene analysis for reducing motion jpeg 2000 video surveillance delivery bandwidth and complexity. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 1, pages I–577. IEEE.

Mitchell, H. B. (2007). *Multi-sensor data fusion*. Springer.

Mittal, A. and Davis, L. S. (2003). M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal of Computer Vision*, 51(3):189–203.

Molina, J. M., Herrero, J. G., Jimenez, F. J., and Casar, J. R. (2004). Fuzzy reasoning in a multiagent system of surveillance sensors to manage cooperatively the sensor-to-task assignment problem. *Applied Artificial Intelligence*, 18(8):673–711.

Molina Lopez, J., Jimenez Rodriguez, S., and Corredera, J. (1995). Fuzzy reasoning for multisensor management. In *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, volume 2, pages 1398–1403. IEEE.

Molina López, J. M., García Herrero, J., Jiménez Rodríguez, F. J., and Casar Corredera, J. R. (2003). Cooperative management of a net of intelligent surveillance agent sensors. *International Journal of Intelligent Systems*, 18(3):279–307.

Monari, E., Voth, S., and Kroschel, K. (2008). An object-and task-oriented architecture for automated video surveillance in distributed sensor networks. In *Advanced Video and Signal Based Surveillance, 2008. AVSS'08. IEEE Fifth International Conference on*, pages 339–346. IEEE.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial & Applied Mathematics*, 5(1):32–38.

Murphy, M. N. (2013). *Contemporary piracy and maritime terrorism: the threat to international security*. Routledge.

Murray, A. T., Kim, K., Davis, J. W., Machiraju, R., and Parent, R. (2007). Coverage optimization to support security monitoring. *Computers, Environment and Urban Systems*, 31(2):133–147.

Musick, S. and Malhotra, R. (1994). Chasing the elusive sensor manager. In *Aerospace and Electronics Conference, 1994. NAECON 1994., Proceedings of the IEEE 1994 National*, pages 606–613. IEEE.

Naman, A. T. and Taubman, D. (2007a). A novel paradigm for optimized scalable video transmission based on jpeg2000 with motion. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 5, pages V–93. IEEE.

Naman, A. T. and Taubman, D. (2007b). A novel paradigm for optimized scalable video transmission based on jpeg2000 with motion. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 5, pages V–93. IEEE.

Nandhakumar, N. and Aggarwal, J. K. (1988). Integrated analysis of thermal and visual images for scene interpretation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(4):469–481.

Natarajan, P., Hoang, T. N., Low, K. H., and Kankanhalli, M. (2012a). Decision-theoretic approach to maximizing observation of multiple targets in multi-camera surveillance. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 155–162. International Foundation for Autonomous Agents and Multiagent Systems.

Natarajan, P., Hoang, T. N., Low, K. H., and Kankanhalli, M. (2012b). Decision-theoretic coordination and control for active multi-camera surveillance in uncertain, partially observable environments. In *Distributed Smart Cameras (ICDSC), 2012 Sixth International Conference on*, pages 1–6. IEEE.

Ng, G. W. and Ng, K. H. (2000). Sensor management–what, why and how. *Information Fusion*, 1(2):67–75.

Nilsson, M. (2007). Human decision making and information fusion: Extending the concept of decision support.

Nilsson, M. (2008). Mind the gap: human decision making and information fusion.

Otsason, V., Varshavsky, A., LaMarca, A., and De Lara, E. (2005). Accurate gsm indoor localization. In *UbiComp 2005: Ubiquitous Computing*, pages 141–158. Springer.

Ozyildiz, E., Krahnstöver, N., and Sharma, R. (2002). Adaptive texture and color segmentation for tracking moving objects. *Pattern recognition*, 35(10):2013–2029.

Pei, L., Chen, R., Liu, J., Kuusniemi, H., Tenhunen, T., and Chen, Y. (2010). Using inquiry-based bluetooth rssi probability distributions for indoor positioning. *Journal of Global Positioning Systems*, 9(2):122–130.

Pennebaker, W. B. and Mitchell, J. L. (1993). *JPEG: Still image data compression standard*. Springer Science & Business Media.

Perera, C., Zaslavsky, A., Christen, P., Compton, M., and Georgakopoulos, D. (2013). Context-aware sensor search, selection and ranking model for internet of things middleware. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 1, pages 314–322. IEEE.

Pérez, P., Hue, C., Vermaak, J., and Gangnet, M. (2002). Color-based probabilistic tracking. In *Computer vision—ECCV 2002*, pages 661–675. Springer.

Petrov, P., Boumbarov, O., and Muratovski, K. (2008). Face detection and tracking with an active camera. In *Intelligent Systems, 2008. IS'08. 4th International IEEE Conference*, volume 2, pages 14–34. IEEE.

Piccardi, M. (2004). Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE.

Plehn, M. T. (2000). Control warfare: Inside the ooda loop. Technical report, DTIC Document.

Possegger, H., Ruther, M., Sternig, S., Mauthner, T., Klopschitz, M., Roth, P., and Bischof, H. (2012). Unsupervised calibration of camera networks and virtual ptz cameras. *17th Computer Vision Winter Workshop*.

Puwein, J., Ziegler, R., Ballan, L., and Pollefeys, M. (2012). Ptz camera network calibration from moving people in sports broadcasts. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 25–32. IEEE.

Qureshi, F. and Terzopoulos, D. (2008). Smart camera networks in virtual reality. *Proceedings of the IEEE*, 96(10):1640–1656.

Rasmussen, C. and Hager, G. D. (2001). Probabilistic data association methods for tracking complex visual objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(6):560–576.

Russell, S., Norvig, P., and Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25.

Saaty, T. L. (1990). How to make a decision: the analytic hierarchy process. *European journal of operational research*, 48(1):9–26.

Saaty, T. L. (2003). Decision-making with the ahp: Why is the principal eigenvector necessary. *European journal of operational research*, 145(1):85–91.

Santa-Cruz, D. and Ebrahimi, T. (2000). An analytical study of jpeg 2000 functionalities. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 2, pages 49–52. IEEE.

Santa-Cruz, D., Ebrahimi, T., Askelof, J., Larsson, M., and Christopoulos, C. A. (2000). Jpeg 2000 still image coding versus other standards. In *International Symposium on Optical Science and Technology*, pages 446–454. International Society for Optics and Photonics.

Schmaedeke, W. W. (1993). Information-based sensor management. In *Optical Engineering and Photonics in Aerospace Sensing*, pages 156–164. International Society for Optics and Photonics.

Schmaedeke, W. W. and Kastella, K. D. (1998). Information-based sensor management and immkf. In *Aerospace/Defense Sensing and Controls*, pages 390–401. International Society for Optics and Photonics.

Schwehr, K. D. and McGillivary, P. A. (2007). *Marine Ship Automatic Identification System (AIS) for enhanced coastal security capabilities: an oil spill tracking application*. IEEE.

Shen, D., Chen, G., Pham, K., and Blasch, E. (2011). A trust-based sensor allocation algorithm in cooperative space search problems. In *SPIE Defense, Security, and Sensing*, pages 80440C–80440C. International Society for Optics and Photonics.

Shi, Y. Q. and Sun, H. (1999). *Image and video compression for multimedia engineering: fundamentals, algorithms, and standards*. CRC press.

Shibata, M., Yasuda, Y., and Ito, M. (2008). Moving Object Detection for Active Camera based on Optical Flow Distortion. *Electrical Engineering*, pages 14720–14725.

Shirai, D., Yamaguchi, T., Shimizu, T., Murooka, T., and Fujii, T. (2006). 4k shd real-time video streaming system with jpeg 2000 parallel codec. In *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, pages 1855–1858. IEEE.

Shoham, Y. (1993). Agent-oriented programming. *Artificial intelligence*, 60(1):51–92.

Shotton, J., Blake, A., and Cipolla, R. (2005). Contour-based learning for object detection. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 503–510. IEEE.

Shulsky, A. N. and Schmitt, G. J. (2002). *Silent warfare: understanding the world of intelligence*. Potomac Books, Inc.

Singh, S., Saini, A. K., Saini, R., Mandal, A., Shekhar, C., and Vohra, A. (2012). Real-time video acquisition and ptz camera movement control for fpga based automated video surveillance system. *International Journal of Research & Reviews in Computer Science*, 3(2).

Singh, V. K., Atrey, P. K., and Kankanhalli, M. S. (2008). Coopetitive multi-camera surveillance using model predictive control. *Machine Vision and applications*, 19(5-6):375–393.

Skodras, A. N., Christopoulos, C. A., and Ebrahimi, T. (2001). Jpeg2000: The upcoming still image compression standard. *Pattern Recognition Letters*, 22(12):1337–1345.

Smith, J., Rhyne, R., et al. (2000). A fuzzy logic resource manager and underlying data mining techniques. In *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, volume 2, pages WEB1–3. IEEE.

Smith III, J. F. and Rhyne II, R. D. (1999). A fuzzy logic algorithm for optimal allocation of distributed resources. In *Fusion 99: Proceednings of the Second International Conference on Information Fusion*, pages 402–409.

Snidaro, L., Foresti, G., Niu, R., and Varshney, P. (2004). Sensor fusion for video surveillance.

Snidaro, L., Niu, R., Varshney, P. K., and Foresti, G. L. (2003). Automatic camera selection and fusion for outdoor surveillance under changing weather conditions. In *Advanced Video and Signal Based Surveillance, 2003. Proceedings. IEEE Conference on*, pages 364–369. IEEE.

Song, B., Ding, C., Kamal, A. T., Farrell, J. A., and Roy-Chowdhury, A. K. (2011). Distributed camera networks. *Signal Processing Magazine, IEEE*, 28(3):20–31.

Song, B., Soto, C., Roy-Chowdhury, A. K., and Farrell, J. A. (2008). Decentralized camera network control using game theory. In *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pages 1–8. IEEE.

Soto, C., Song, B., and Roy-Chowdhury, A. K. (2009). Distributed multi-target tracking in a self-configuring camera network. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1486–1493. IEEE.

Spaan, M. T. and Lima, P. U. (2009). Decision-theoretic planning under uncertainty for cooperative active perception.

Spezio, A. E. (2002). Electronic warfare systems. *Microwave Theory and Techniques, IEEE Transactions on*, 50(3):633–644.

Systems, F. (2009). Flir systems thermal imaging cameras for coastal surveillance applications. Press Release. http://www.flir.com/uploadedfiles/cs_emea/application_stories/media/downloads/coastalsurv_en.pdf.

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer.

Tarabanis, K. A., Allen, P. K., and Tsai, R. Y. (1995). A survey of sensor planning in computer vision. *Robotics and Automation, IEEE Transactions on*, 11(1):86–104.

Ukita, N. and Matsuyama, T. (2002). Real-time multi-target tracking by cooperative distributed active vision agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, pages 829–838. ACM.

Ukita, N. and Matsuyama, T. (2005). Real-time cooperative multi-target tracking by communicating active vision agents. *Computer Vision and Image Understanding*, 97(2):137–179.

Valera, M. and Velastin, S. (2005). Intelligent distributed surveillance systems: a review. In *Vision, Image and Signal Processing, IEE Proceedings-*, volume 152, pages 192–204. IET.

Van Droogenbroeck, M. and Talbot, H. (1996). Fast computation of morphological operations with arbitrary structuring elements. *Pattern recognition letters*, 17(14):1451–1460.

Varma, K. and Bell, A. (2004). Jpeg2000-choices and tradeoffs for encoders. *Signal Processing Magazine, IEEE*, 21(6):70–75.

Wallace, G. K. (1991). The jpeg still picture compression standard. *Communications of the ACM*, 34(4):30–44.

Wang, G., Cao, G., and La Porta, T. (2006). Movement-assisted sensor deployment. *Mobile Computing, IEEE Transactions on*, 5(6):640–652.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612.

Watson, A. B. (1998). Toward a perceptual video-quality metric. In *Photonics West'98 Electronic Imaging*, pages 139–147. International Society for Optics and Photonics.

Watson, A. B., Hu, J., and McGowan, J. F. (2001). Digital video quality metric based on human vision. *Journal of Electronic imaging*, 10(1):20–29.

Wesson, R., Hayes-Roth, F., Burge, J., Stasz, C., and Sunshine, C. (1981). Network structures for distributed situation assessment. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(1):5–23.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(02):115–152.

Wu, G., Wu, Y., Jiao, L., Wang, Y.-F., and Chang, E. Y. (2003). Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 528–538. ACM.

Xia, F., Zhao, W., Sun, Y., and Tian, Y.-C. (2007). Fuzzy logic control based qos management in wireless sensor/actuator networks. *Sensors*, 7(12):3179–3191.

Xiao, F. et al. (2000). Dct-based video quality evaluation. *Final Project for EE392J*, 769.

Xiong, N. and Svensson, P. (2002). Multi-sensor management for information fusion: issues and approaches. *Information fusion*, 3(2):163–186.

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13.

Yilmaz, A., Li, X., and Shah, M. (2004). Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1531–1536.

Zhang, Z. (2000). A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334.

Zhao, F., Liu, J., Liu, J., Guibas, L., and Reich, J. (2003). Collaborative signal and information processing: an information-directed approach. *Proceedings of the IEEE*, 91(8):1199–1209.

Zhao, F., Shin, J., and Reich, J. (2002). Information-driven dynamic sensor collaboration. *Signal Processing Magazine, IEEE*, 19(2):61–72.

Zou, Y. and Chakrabarty, K. (2003). Sensor deployment and target localization based on virtual forces. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1293–1303. IEEE.